iPhone Project

Bill Qualls

DePaul University – Spring 2011

ECT 587 Mobile Commerce Technology

Professor Eugene Backlin

Contents

- Introduction
- Welcome
- Main Menu
- Monthly Payment Calculator
- Showcase Of Homes
- Our Convenient Offices
- Our Friendly Agents
- SQLite
- Lessons Learned

INTRODUCTION

Introduction

- This is my first iPhone app!
- I developed this app for a friend of mine.
- He is the owner of Kettley Realtors, the largest realtor company in the Fox Valley.
- This is my attempt at giving Kettley an iPhone presence.
- I envision this app being distributed freely to anyone who wants it.
- I think it has the potential for increasing traffic from buyers and sellers alike.

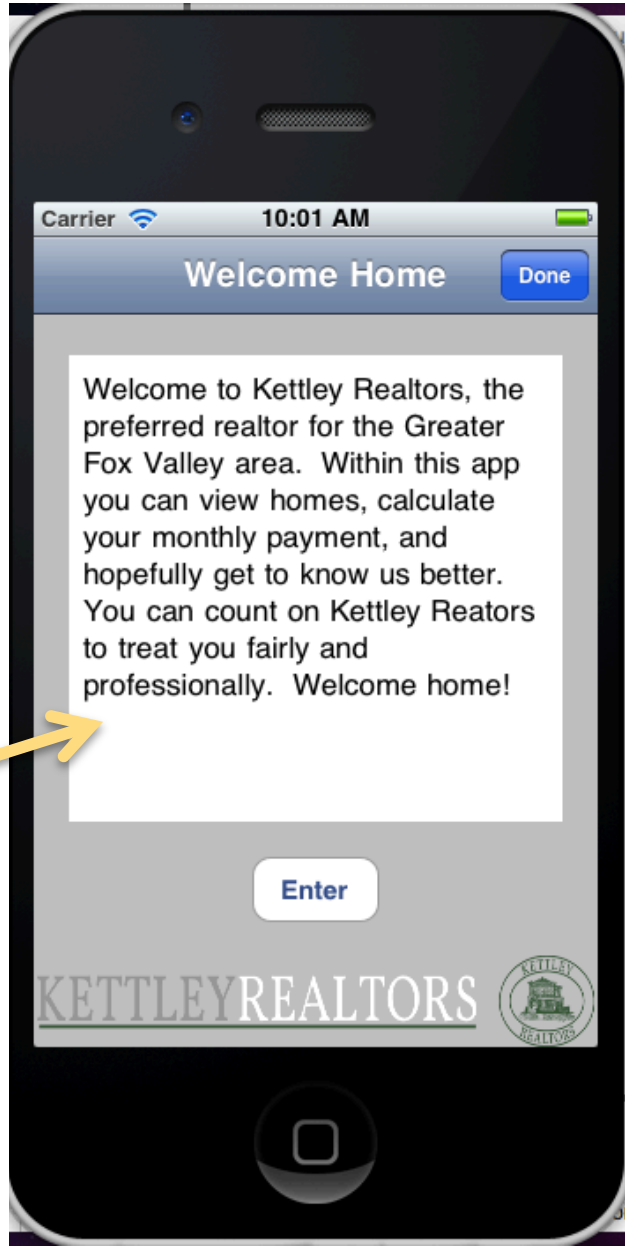
Introduction

- This presentation will show the functionality of the app.
- But the primary purpose of this presentation is to highlight the features of Xcode which I used in creating the app.
- With any luck, it will be sufficient to help me remember how I did what I did!
- As such, it will include Xcode and SQLite snippets.

WELCOME

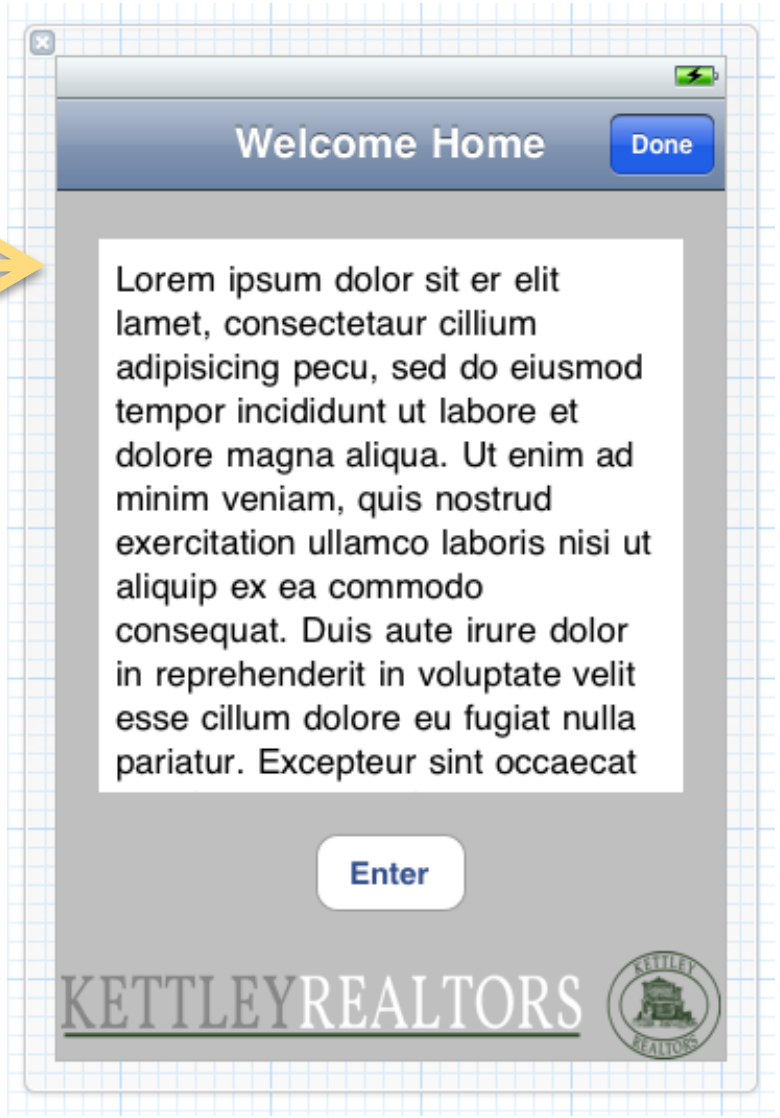
Welcome

- The Welcome screen provides my friend with an opportunity to say a few words…
- Xcode features:
  - Text area populated with text read from a file.



Welcome

- The Welcome xib

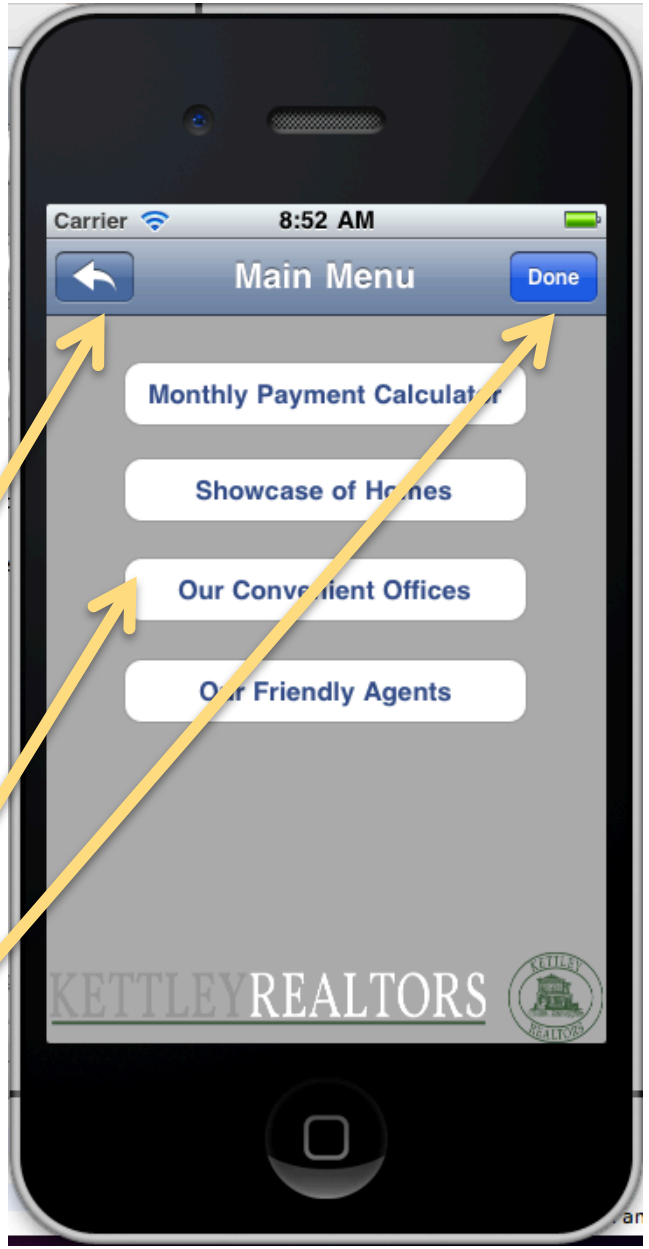


- XIB = Xcode Interface Builder
- XIB was preceeded by NextStep Interface Builder, so it is usually pronounced "nib".

MAIN MENU

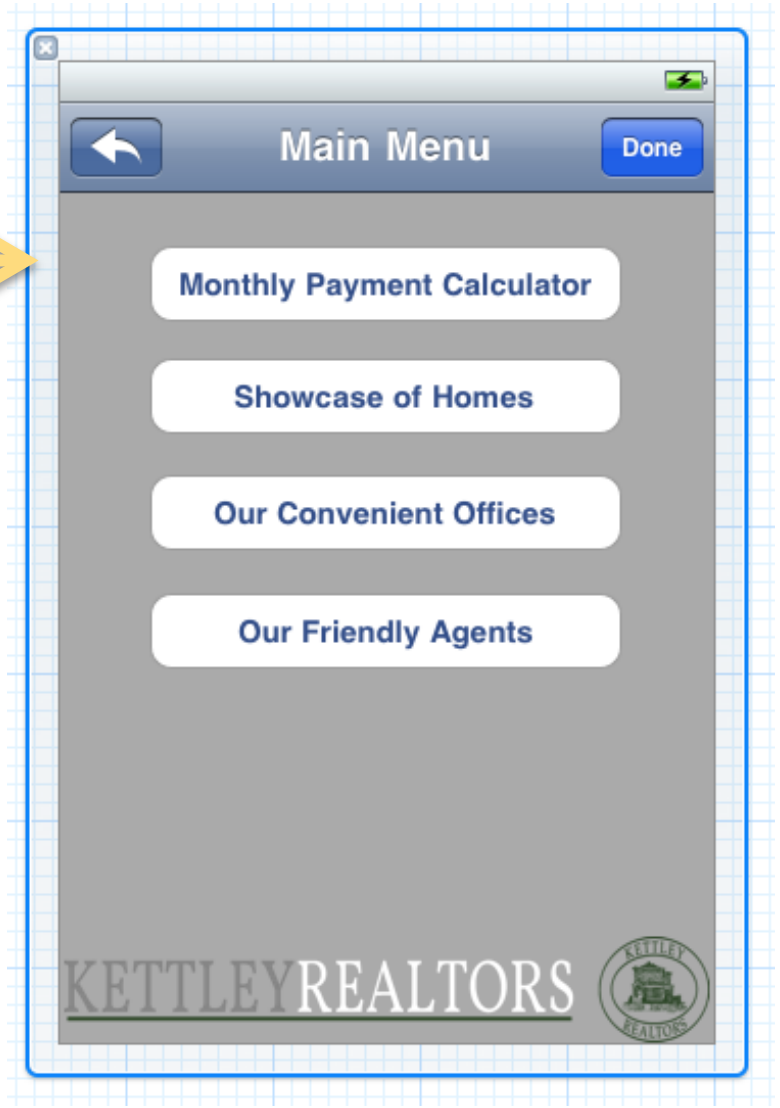
Main Menu

- The Main Menu provides access to the app's features…
- Xcode features:
  - Back button returns to greeting page
  - Buttons provide access to features
  - Done button exits app



Main Menu xib

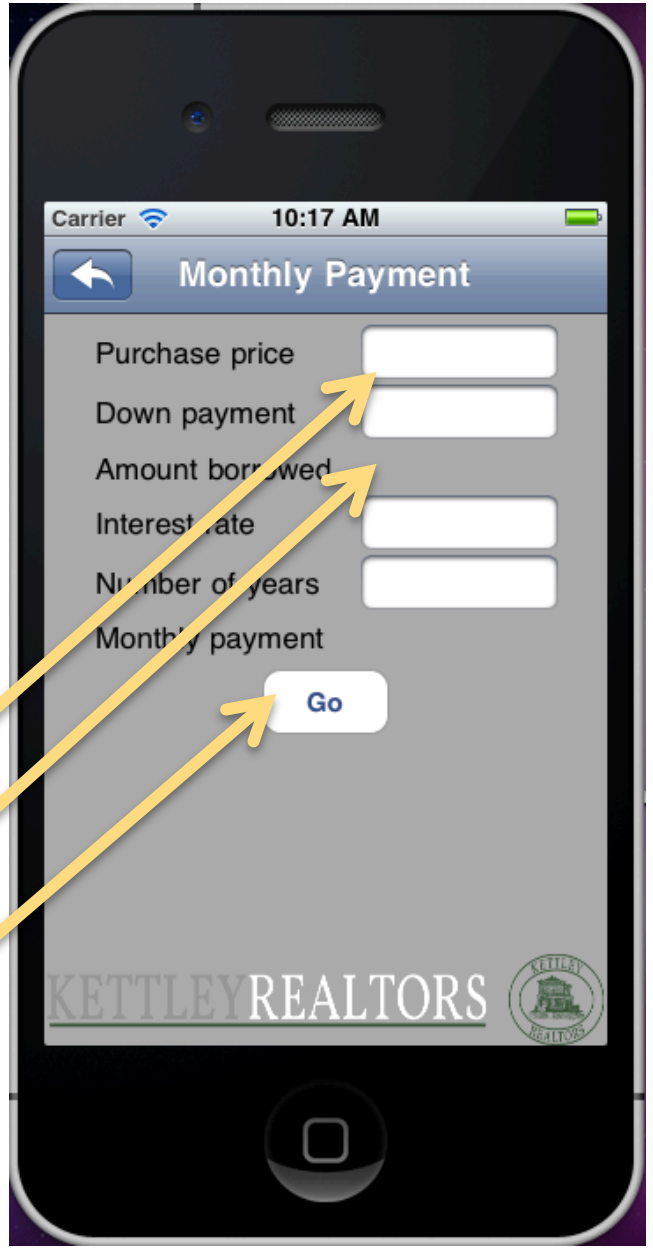
- The main menu xib



MONTHLY PAYMENT CALCULATOR

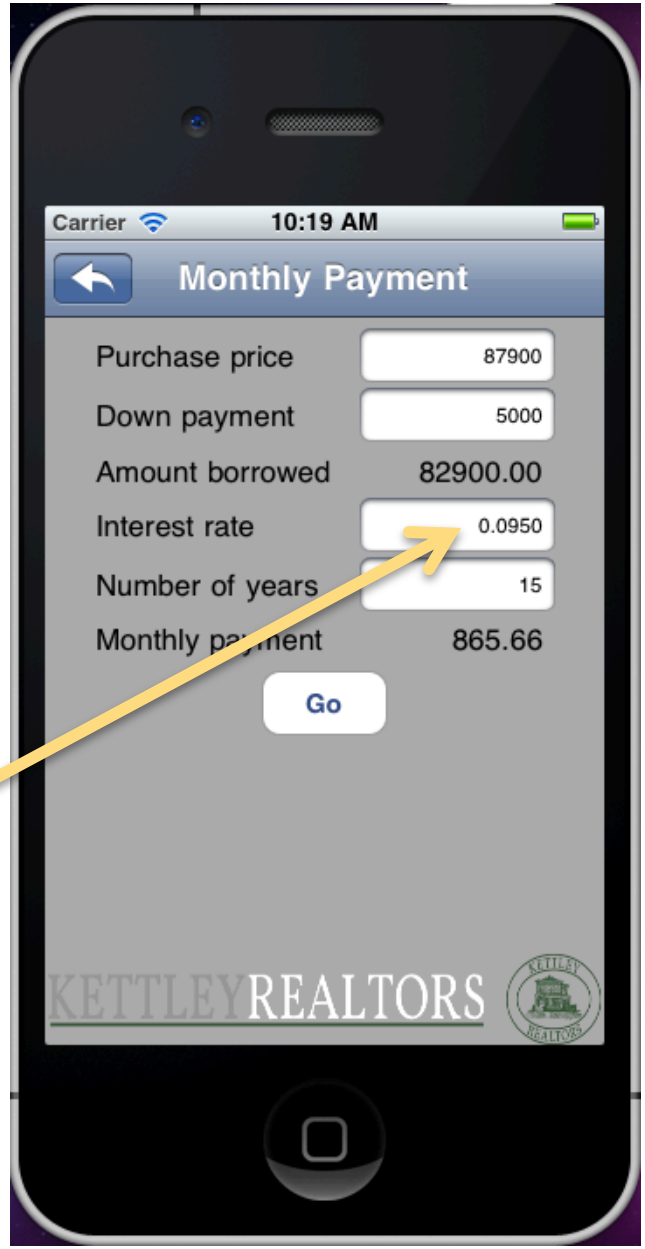
Calculator

- The Monthly Payment Calculator seemed a natural addition to a real estate app…
- This was the first feature I completed.
- Xcode features:
  - Text fields for inputs
  - Labels for prompts and outputs
  - Button to calculate



Calculator

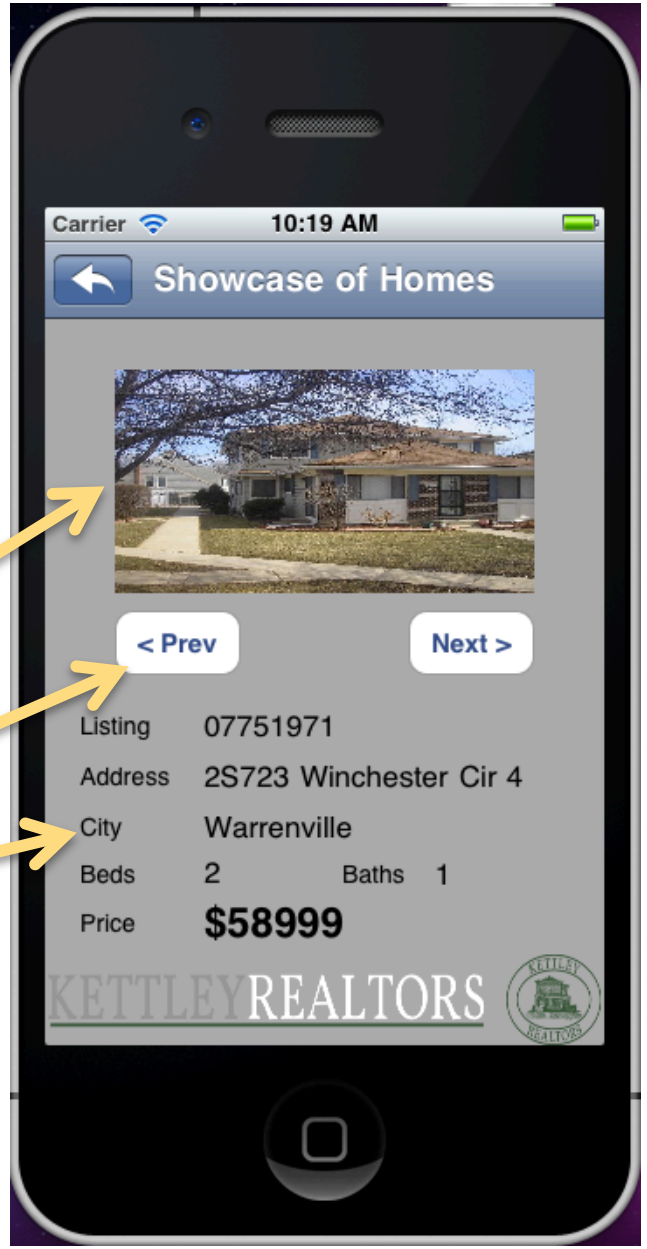
- Sample usage
- If user enters an interest rate greater than or equal to one (example: 9.5), then it is automatically divided by 100 and displayed as such (example: 0.0950 as shown here.)



SHOWCASE OF HOMES

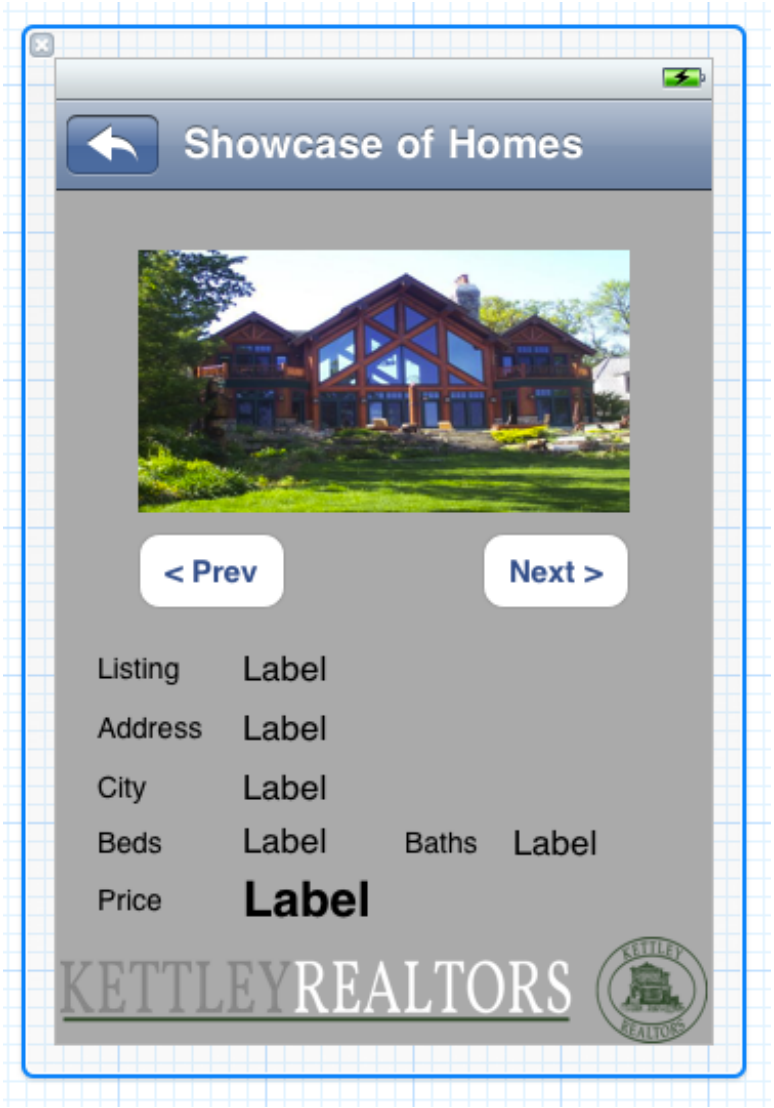
Homes

- The Homes screen allows the user to browse current listings…
- Xcode features:
  - Showing an image
  - Prev and Next buttons
  - Data from an SQLite database



Homes xib

- The Homes xib file…



Home.h

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

@interface Home : NSObject
{
    NSString *listing;
    NSString *address;
    NSString *city;
    NSString *zip;
    NSNumber *price;
    NSNumber *bedrooms;
    NSNumber *bathrooms;
}

@property (nonatomic, retain) NSString *listing;
@property (nonatomic, retain) NSString *address;
@property (nonatomic, retain) NSString *city;
@property (nonatomic, retain) NSString *zip;
@property (nonatomic, retain) NSNumber *price;
@property (nonatomic, retain) NSNumber *bedrooms;
@property (nonatomic, retain) NSNumber *bathrooms;

-(id)initWithListing:(NSString *)l address:(NSString *)a
city:(NSString *)c zip:(NSString *)z price:(NSNumber *)p
bedrooms:(NSNumber *)bd bathrooms:(NSNumber *)ba;

@end
```

Home.m

```
#import "Home.h"

@implementation Home

@synthesize listing;
@synthesize address;
@synthesize city;
@synthesize zip;
@synthesize price;
@synthesize bedrooms;
@synthesize bathrooms;

-(id)initWithListing:(NSString *)l address:(NSString *)a
city:(NSString *)c zip:(NSString *)z price:(NSNumber *)p
bedrooms:(NSNumber *)bd bathrooms:(NSNumber *)ba
{
    self.listing = l;
    self.address = a;
    self.city = c;
    self.zip = z;
    self.price = p;
    self.bedrooms = bd;
    self.bathrooms = ba;
    return self;
}

@end
```

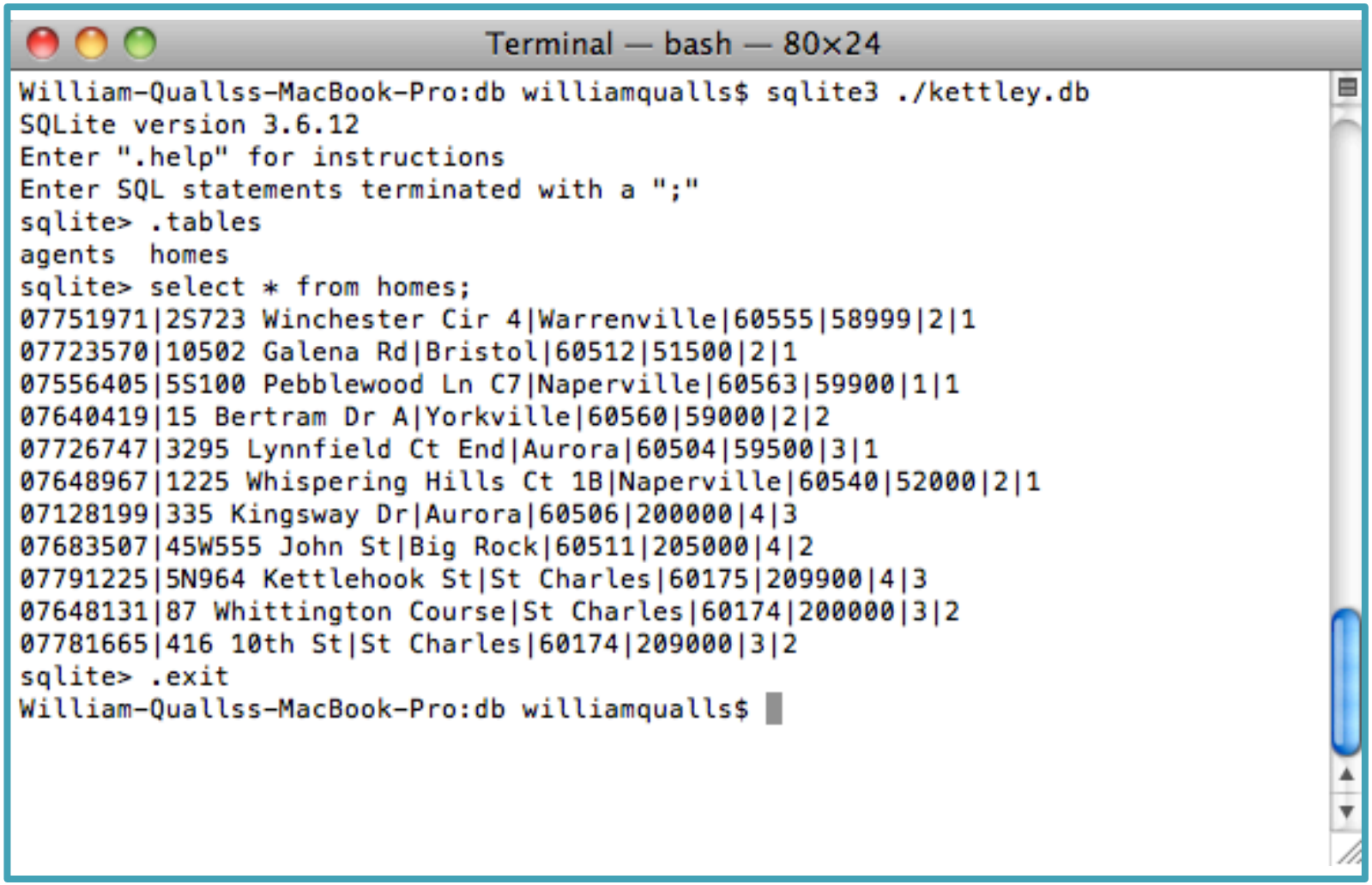
SQLite: Homes table

Open command window. Navigate to desired folder (use `cd` and `ls` commands as necessary)

```
williamqualls$ sqlite3 ./kettley.db
sqlite> create table homes (listing text primary key, address text, city text, zip text, price integer, bedrooms smallint, bathrooms smallint);
sqlite> insert into homes values ("07751971", "2S723 Winchester Cir 4", "Warrenville", "60555", 58999, 2, 1);
sqlite> insert into homes values ("07723570", "10502 Galena Rd", "Bristol", "60512", 51500, 2, 1);
```

(etc.)

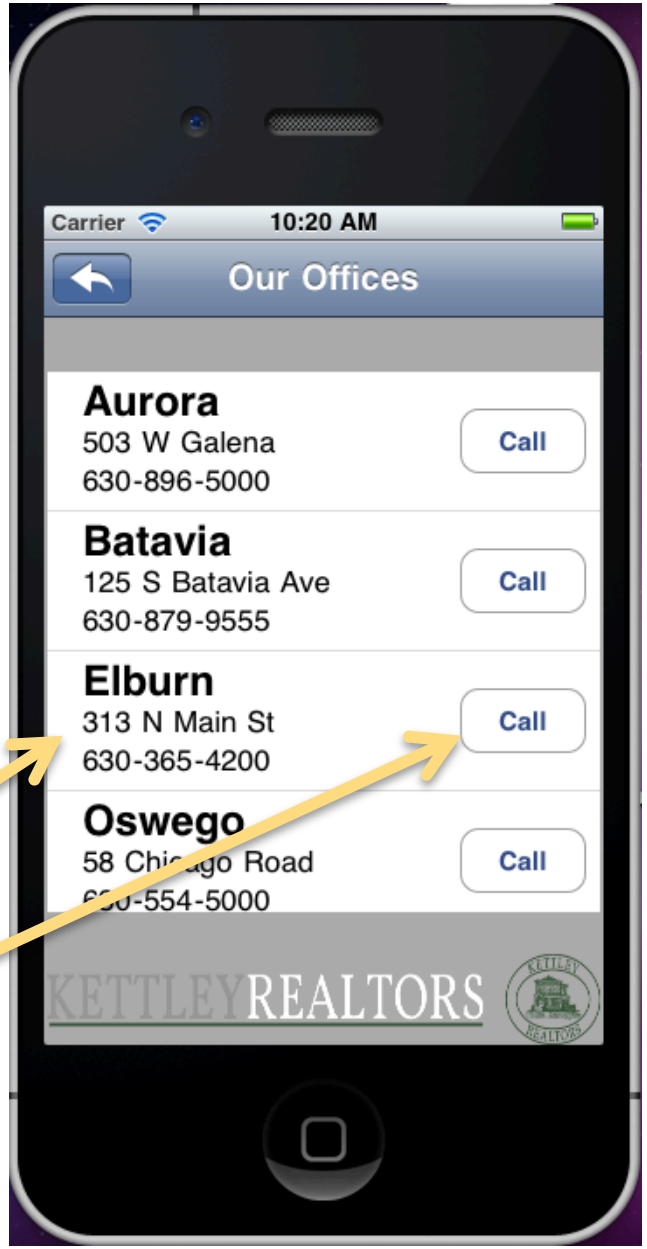
SQLite: Homes table



OUR CONVENIENT OFFICES

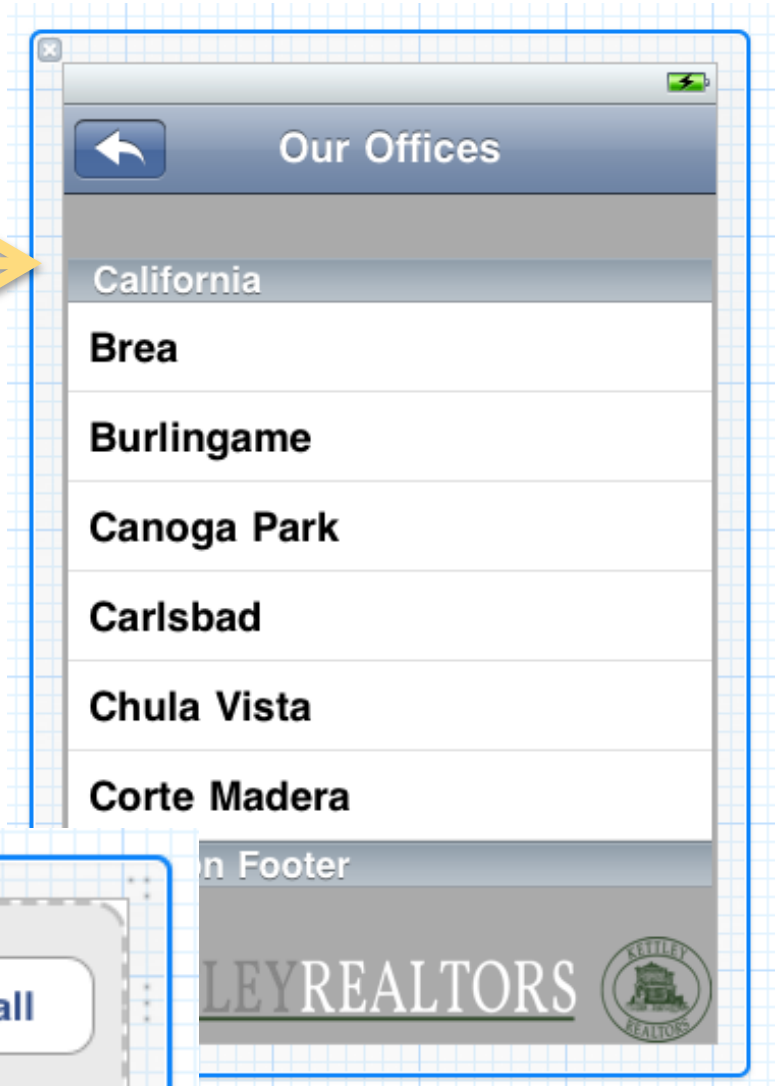
Offices

- The Offices screen provides quick access to office addresses and phone numbers…
- Xcode features:
  - Table view
  - UITableView with custom table view cell
  - Push button to dial phone
  - Data populated from a Property List (plist)



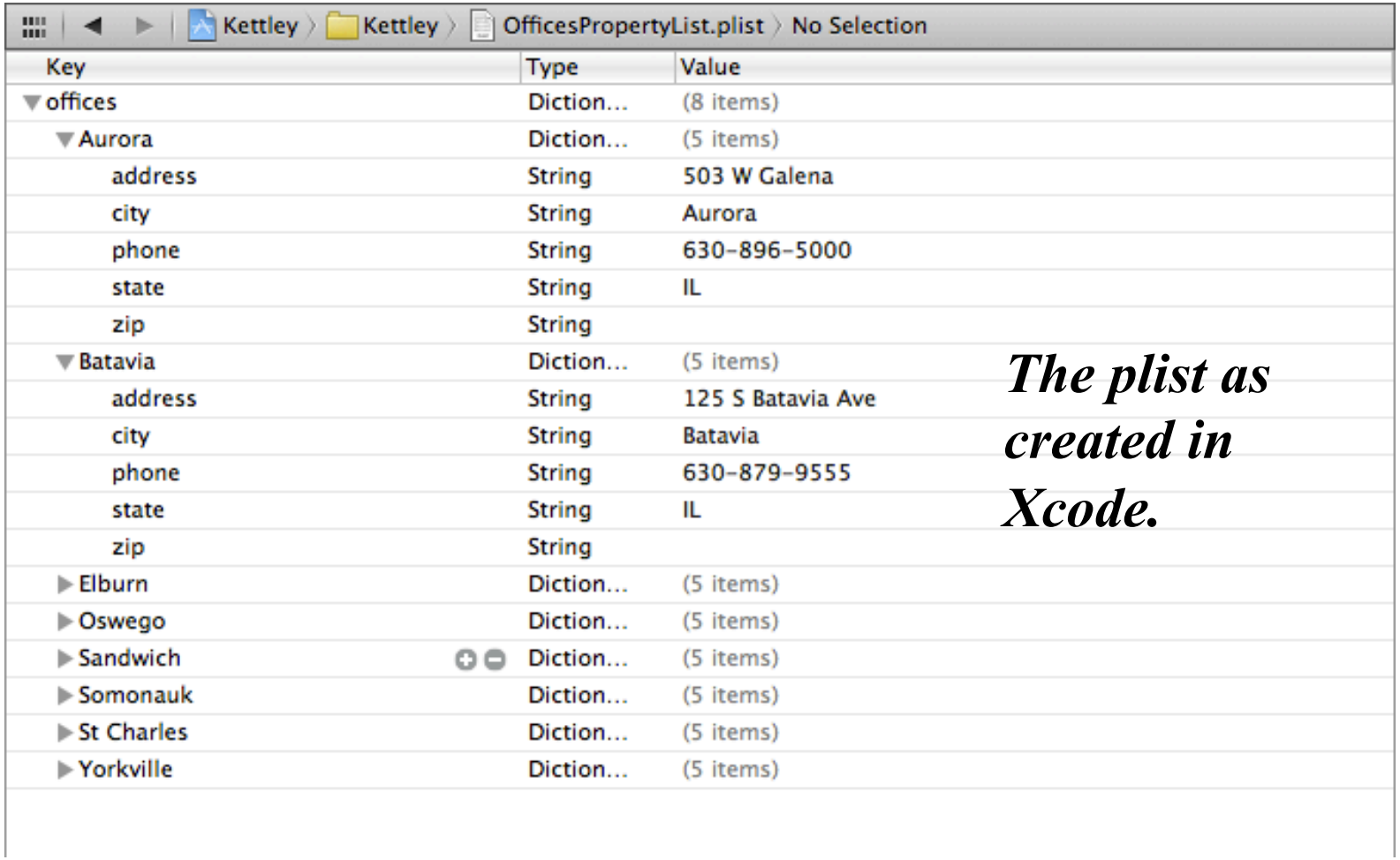
Offices xib

- The Offices xib file



- …and the custom table view cell

OfficesPropertyList.plist



The plist as created in Xcode.

OfficesPropertyList.plist

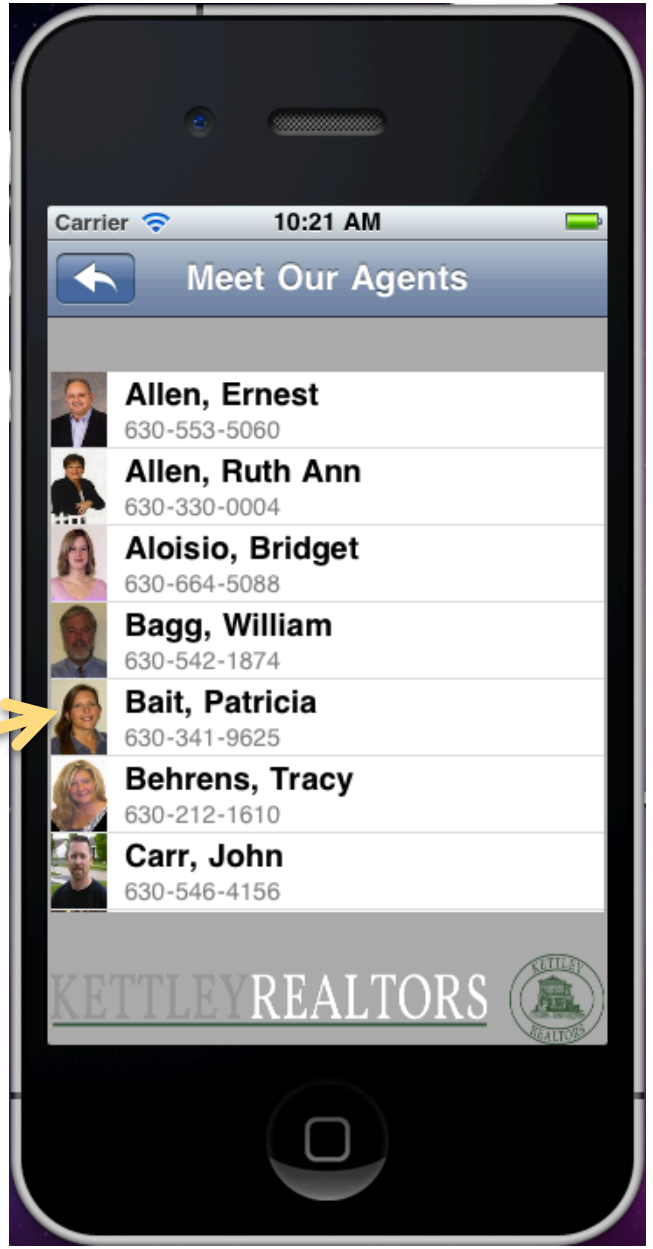
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/
DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>offices</key>
        <dict>
        <key>Aurora</key>
        <dict>
            <key>address</key>
            <string>503 W Galena</string>
            <key>city</key>
            <string>Aurora</string>
            <key>phone</key>
            <string>630-896-5000</string>
            <key>state</key>
            <string>IL</string>
            <key>zip</key>
            <string></string>
        </dict>
        <key>Batavia</key>
        <dict>
            <key>address</key>
            <string>125 S Batavia Ave</string>
```

As you can see, a plist is stored as an XML file.

OUR FRIENDLY AGENTS

Agents

- The Agents screen shows the name, phone number, and photo of all agents…
- Xcode features:
  - UITableView with UITableViewCellStyleSubtitle
  - Data from an SQLite database



Agent.h

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

@interface Agent : NSObject
{
    NSString *lastname;
    NSString *firstname;
    NSString *phone;
    NSString *image;
}

@property (nonatomic, retain) NSString *lastname;
@property (nonatomic, retain) NSString *firstname;
@property (nonatomic, retain) NSString *phone;
@property (nonatomic, retain) NSString *image;

-(id)initWithLastname:(NSString *)l firstname:(NSString *)f
phone:(NSString *)p image:(NSString *)i;

@end
```

instance variables

constructor

Agent.m

```
#import "Agent.h"

@implementation Agent

@synthesize lastname;
@synthesize firstname;
@synthesize phone;
@synthesize image;

-(id)initWithLastname:(NSString *)l firstname:(NSString *)f
phone:(NSString *)p image:(NSString *)i
{
    self.lastname = l;
    self.firstname = f;
    self.phone = p;
    self.image = i;
    return self;
}

@end
```

Agents: SQLite table

Open command window. Navigate to desired folder (use `cd` and `ls` commands as necessary)

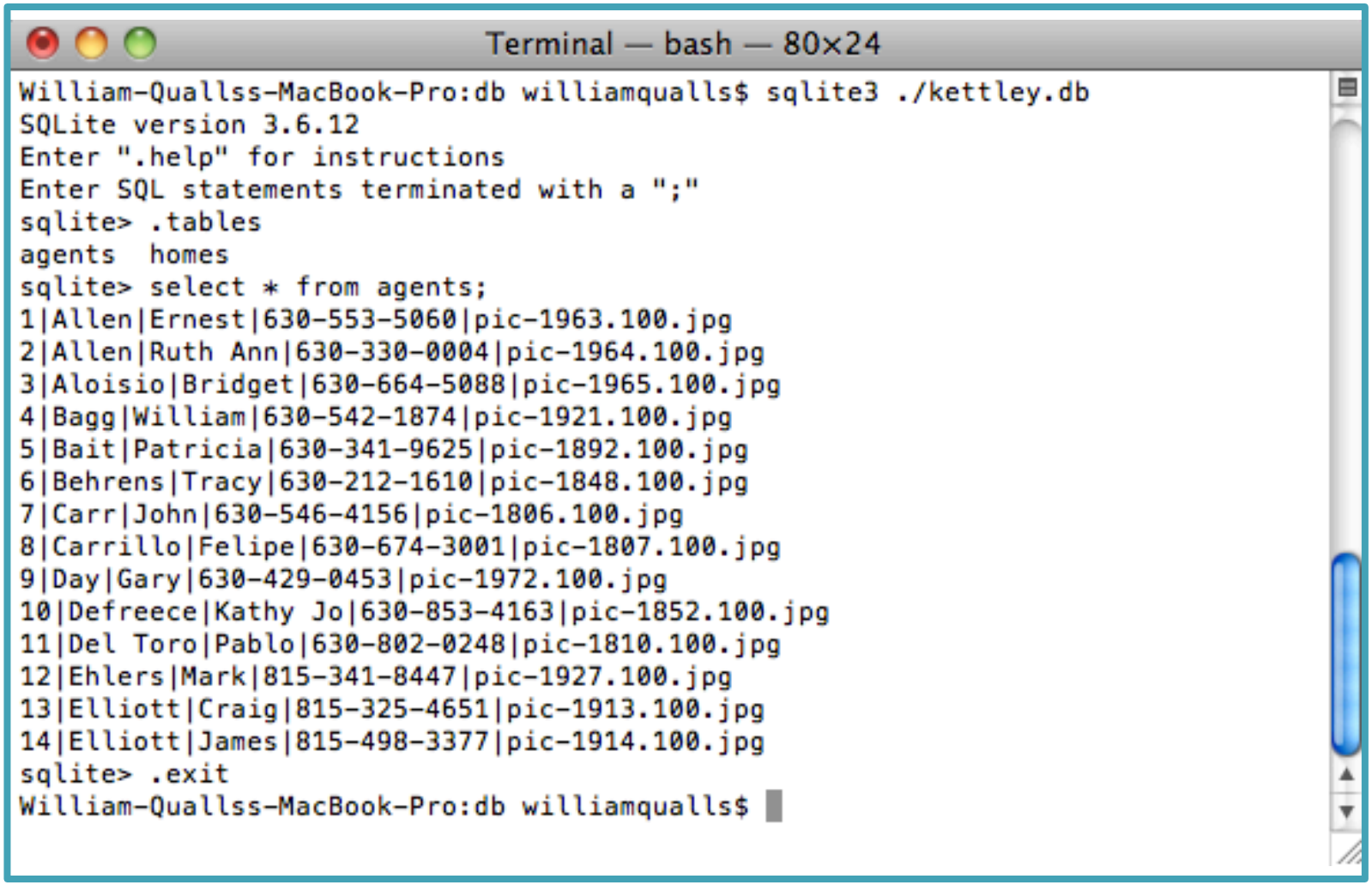
```
williamqualls$ sqlite3 ./kettley.db
sqlite> create table agents(id integer primary key autoincrement, lastname text, firstname text, phone text, image text);
sqlite> insert into agents (lastname, firstname, phone, image) values ("Allen", "Ernest", "630-553-5060", "pic-1963.100.jpg");
sqlite> insert into agents (lastname, firstname, phone, image) values ("Allen", "Ruth Ann", "630-330-0004", "pic-1964.100.jpg");
```

(etc.)

Agents: SQLite table

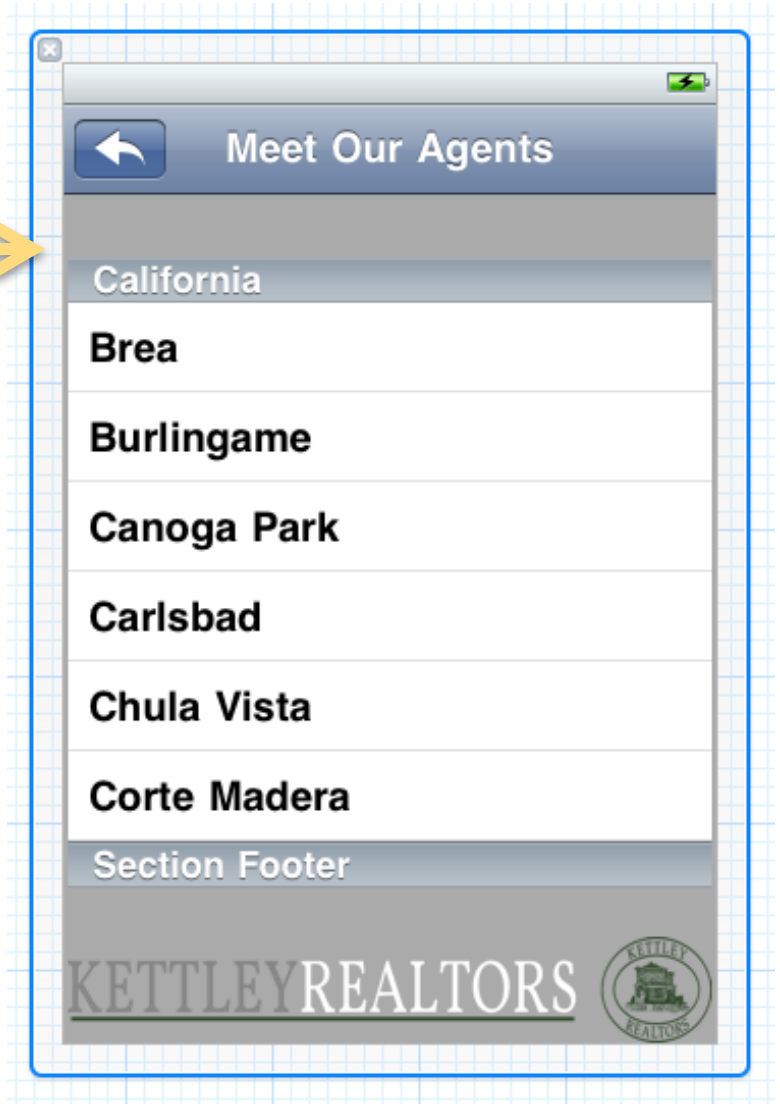
- The Agents table uses "id integer primary key autoincrement" to create a unique identifier.
- This field is not included on an "insert" statement.
- …but it is returned in response to a "select *" statement.

Agents: SQLite table



Agents xib

- The Agents xib file



- Using UITableViewCell StyleSubtitle enables display of:
  - textLabel (name)
  - detailTextLabel (phone)
  - imageView (photo)

UITableViewCellStyleSubtitle

```
Agent *aAgent = (Agent *)[self.agents objectAtIndex:[indexPath
row]];

UIImage *cellImage = [UIImage imageNamed:[aAgent image]];
cell.imageView.image = cellImage;

NSString *fullname = [[aAgent lastname]
            stringByAppendingString:@", "];
fullname = [fullname stringByAppendingString:[aAgent
firstname]];
cell.textLabel.text=fullname;

cell.detailTextLabel.text = [aAgent phone];
```

SQLITE

SQLite

- As an experienced SQL user, I found the SQLite tool very easy to use.
- As expected, SQL commands end with a semi-colon
- SQLite commands begin with a period and do NOT end with a semi-colon (examples: .tables, .exit)

SQLite: About images

- For homes, I used the listing number, followed by ".jpg", as the image file name.
- For agents, I stored the complete image file name.
- All images were copied to an images folder which was added to the project.

LESSONS LEARNED

Lessons Learned – Mac

- This was my first time using a Mac!
- So some of the things which I note may be obvious to experienced Mac users….
- It was VERY frustrating getting used to it!
- Screen capture: Command + Shift + 4
- Delete key is really a Backspace key: more traditional delete is: Fn + Delete
- Home: Command + Left Arrow
- End: Command + Right Arrow

Lessons Learned – Interface Builder

- This was my biggest source of frustration!
- Despite the capabilites of Xcode 4, I found it easier to define IB fields in the header file first.
- Control drag from File's Owner to XIB for IBOutlet items such as text fields and labels.
- Control drag from XIB to File's Owner for IBAction items such as buttons).

Lessons Learned – Simulator

- I got burned at one point because the simulator was looking at an archived version of the database.
- Symptom: The application wouldn't recognize a new SQLite table.
- To resolve this problem the application must be deleted from the simulator.


Carrier
10:37 PM
Kettley
Safari
Photos


Carrier
10:38 PM
Kettley
Delete “Kettley”
Deleting “Kettley” will also delete all of its data.
Delete
Cancel
Safari
Photos

THANK YOU.