# Principia College

# Engineering Projects: Mobile Apps – ENGR 280 B Fall 2012

# iOS App Development Guide



Figure : *Kishau. Web July 22, 2012. <*[*http://www.kishau.com/wp-content/uploads/2011/03/iphone\_apps.jpg*](http://www.kishau.com/wp-content/uploads/2011/03/iphone_apps.jpg)*>*

Copyright © 2012 Principia College, Departments of Business Administration and Computer Science. All rights reserved.

Table of Contents

iOS App Development Guide 4

1. Apps and the Business of Apps 5

App types and elements 5

Income from apps 6

Marketing 6

Submitting to the app store 6

2. iOS5, Xcode Development Environment 7

Software 7

Loading a sample app: NavBar 7

Parts of the development environment 7

Testing and deploying 8

Debugging: Logs 8

Debugging: Trace 9

3. User Interfaces 11

Navigation 11

Content Screen types 11

Elements 11

Accessing elements in code 12

Selection Elements 16

4. Objects and Objective C 19

Objects 19

Data 20

Initializing 20

Methods 22

Code: processing logic inside a method 22

5. Table views 25

Adding a table to UIViewController 25

Adding an array to hold data to be displayed 27

Responding to cell taps 27

Responding to cell ‘arrow’ tap 28

Creating a custom cell 29

Delete or add cells 30

Headers/footers 30

6. Application Data and Storage 31

iPhone Settings 31

Files 31

Local Database 31

7. Navigation using storyboards 34

8. Integrating with the world 35

RSS Feeds 35

Server database 35

Web services 36

9. Using device tools 40

Orientation 40

Media 40

Email 40

Date formatting 41

Maps and Location 41

Gestures 41

Address book 41

Camera and Photo Library 42

Graphics and Animation 42

Provisioning 42

Distributing on device(s) 42

# iOS App Development Guide

Computing is all about ***interface, data, and process—the action of*** taking user inputs, transforming them and related data and presenting results (output) that is productive to the user. Mobile apps are no different *except* that the user can ask for and gain access to the results--which were perhaps computed on a powerful computer across the world from the user--anywhere, anytime on a device smaller than a book. This, of course, makes mobile devices and their apps the most powerful computing platform in the world. And, oddly, it is not only important to learn how to develop them, its *fun*.

This course will give a concise summary of how to develop iOS apps for Apple’s iPhone and iPad – the most advanced and original app platforms. This guide references sections from the book ***iOS 5 Programming Cookbook* by Vandad Nahavandipoor to** support the exercises and this book is used as a reference book for this course.

**Course Outline**

Week 1: Apps and the Business of Apps, iOS5, xCode development environment, testing in the simulator

Week 2: Interfaces

Week 3: Coding

Week 4-5: Tables

Week 6-7: Application Data

Week 8: Storyboards

Week 9-10: Integrating with the World

Week 11: Device Tools

## 1. Apps and the Business of Apps

End goal: Apple app store? $? Social or technological progress? Business distruption? You decide. Consider: [Business Insider presentation].

Examples: Daily lift app, PracTrak: go to website then iTunes store and show in app store

<https://developer.apple.com/programs/ios/> Don’t have to pay $99 to get dev resources and build apps, just to distribute. Principia has an Enterprise license and can distribute apps.

<https://developer.apple.com/devcenter/ios/index.action>

Click on iTunes connect and view app description and summary.

There are 300,000 apps with 7 billion app downloads on the Apple app store that can be run on 120 million devices (iPhones and iPads). It is the modern day gold rush…it all aspects. Most of the apps and the people producing them can’t make a living off them.

What does it take? The software is free. An idea…a big simple idea that combines *content and communities with anywhere mobile access* …. like Instragram (mobile, social, photos) or mobile, social, and games: DrawSomething.

New ideas fill a gap, solve a problem, or tap into a trend. The power of apps are in connecting multiple ideas and providing power of internet/servers and cloud in a mobile device.

***Mobile***

Anywhere, anytime access

***Content and services***

In-app/local

Streaming audio and video, RSS feeds

Servers: the real power is data in the cloud and extending to servers seemlessly

***Collaborating***

Server-based information sharing

Gaming platform

Peer-to-peer

### App types and elements

#### Navigation

Tabs, Navigation bar, Icon Menu

#### Content

Text, RSS, video, audio, search, WEB Service

#### Graphics

See dev environment app summary screen. Free icons available on the web. (Wooldridge, 92)

### Income from apps

Initial price

Freemium model

In-app purchase

iAds

### Marketing

Brand, Differentiation and 4 P’s (Price, product (app identity), place (App store and community), promotion (social channels!)

Buzz (pre-release social blitz): You-Tube; community for your app; traffic >> web-site via Twitter, Facebook, blogs. Content: Video, screen-shots, description (elevator pitch), cool examples, reviews/testimonials, professional marketing options, Promo codes from Apple (100 free apps), reviews from app sites (Wooldridge, 431).

Submitting an app requires a developer or enterprise license. Apps can be free or charge a fee.

### Submitting to the app store

Submitting to the app store requires a paid developer license and a special build of the app including a provision file for release. The basic steps are as follows and are detailed more in iOS 5 Programming Cookbook pages 16-21.

Step 1) Create an app placeholder and unique identifier within the web portion of the Apple Developer site. Set the app status to ready to upload.

Step 2) Clean and build the app using the iOS Device device setting. Select upload the app.

It sounds simple but all icons must be the right size, settings correct to submit. It will take a week to two weeks for a new app to be approved. Apps are checked for bad language, improper use of internal system calls, really poor design, etc. and are not approved if any of these are found…the app must be solid and practical and ‘clean’. Be careful NOT to use encryption unless you have to as there are special laws regulating deployment of apps to foreign countries with encryption technology in it.

**PROBLEM SET #1: Download University of Texas Austin and iStandford apps. Comment in your journal on structure of one of the apps navigation and use of lists and information. Think of a new feature not seen in the apps you downloaded that you would LOVE to see and would be helpful in a Principia app.**

## iOS5, Xcode Development Environment

I operating system 5 is the base we will use. This was the last major release (as opposed to 3, 4, 4.1, 5.1, etc).

### Software

iOS (iPhone iPad operating system) and XCode for the development environment. All code is Objective C (vs Java-based language for Android)

Free from Apple developer: <https://developer.apple.com/xcode/index.php>

Xcode 4.3.2 runs on Lion.

### Loading a sample app: NavBar

Click the link below and search for NavBar and download the folder for the sample app.

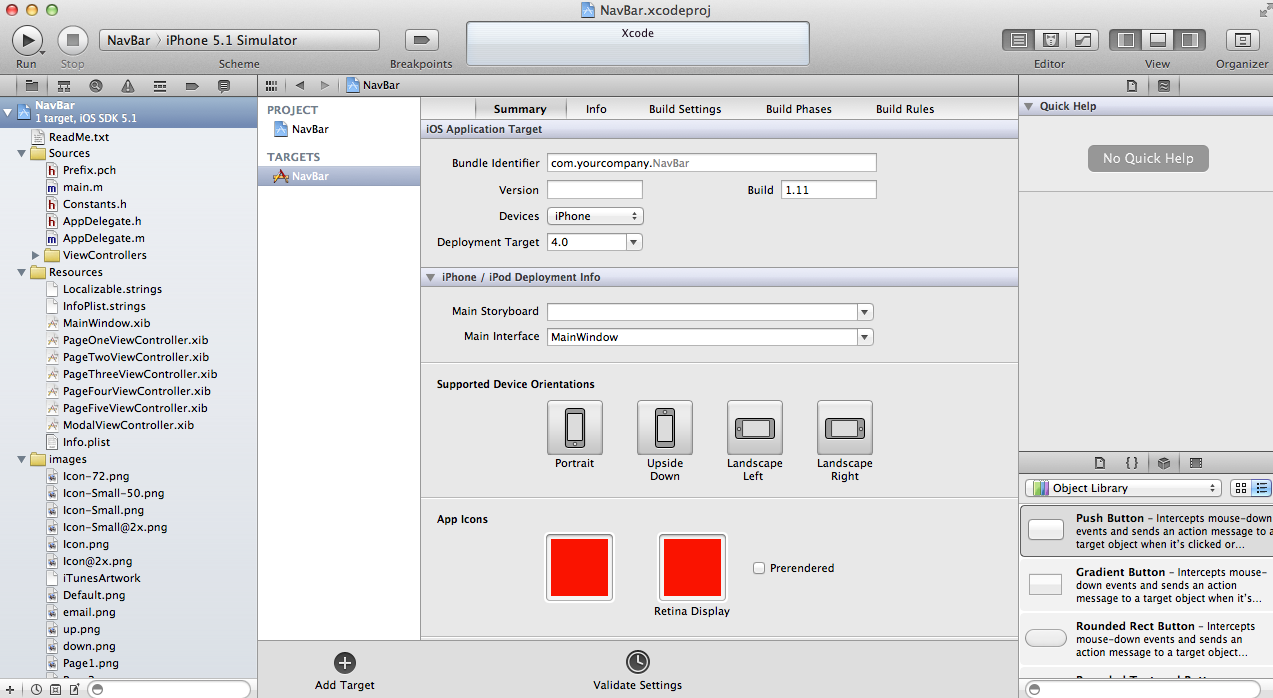
<https://developer.apple.com/library/ios/navigation/#section=Resource%20Types&topic=Sample%20Code>

To open in XCode, double-click navbar.xcodeproj within the NavBar folder.

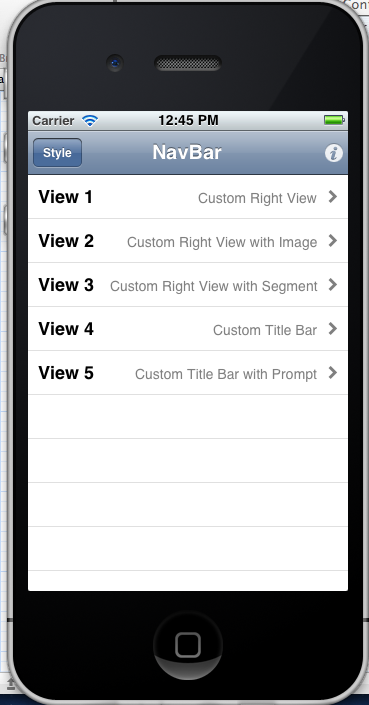
**NOTE: Use samples a lot. Google for code samples and Apple for app samples.**

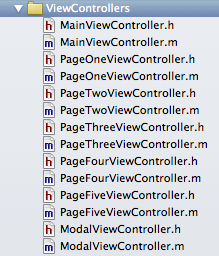
***Google for sample code. Example: “ios date formatting”***

### Parts of the development environment

The developer environment loads as shown below. On the left open NavBar and the sources, resources, and images folders.

#### Sources

List of your object classes. One for each screen, the main program, any custom process/code object classes. Click on AppDelegate.h and .m. This are header and code files. Open the ViewControllers folder and you see the code for all screens in the application.:



.h file is the object class header or object definition

.m is the implementation or code

#### Resources

List of your screens (.xib) files and Images list all icons and images used in the app. Clicking on an .xib file opens the interface builder for that screen. Try it!

#### Running the app in the Simulator

Press Run in upper left with NavBar> iPhone 5.1 Simulator selected next to it. The screen to the right should appear and is an almost fully functioning simulation of the app on your computer.

### Testing and deploying

First, make sure to add error checks in code with appropriate alert messages. The app should be run in the simulator for both phone and iPad (if supported in app) . The log can be used as shown below. Be sure to include logging statements at critical points. Tracing can be used to walk through code and find variable values and bugs.

### Debugging: Logs

The following example code can be used to print out variables in strategic places in the code. Go to the mainviewcontroller.m tableview method and insert the bolded row below the cell.textlable.text… code as follows:

cell.textLabel.text = [dataDictionary valueForKey:kTitleKey];

**NSLog (@"Table text:%@ rows #:%d", [dataDictionary valueForKey:kTitleKey], indexPath.row);**

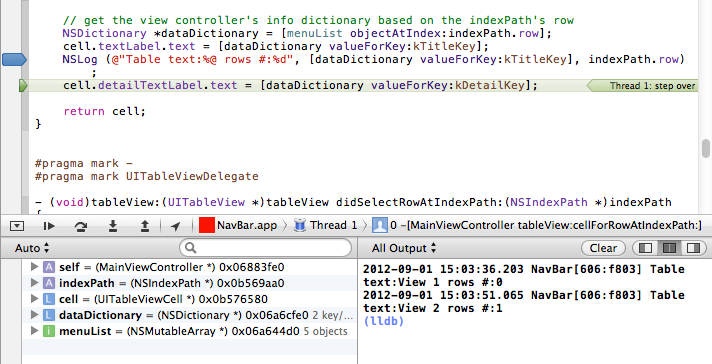
The messages will appear in a bottom log window within the development environment during runtime.

### Debugging: Trace

1. Set breakpoints before sections of code you want to step through. Set trace breakpoints in code by tapping in column to left of code and > appears.

2. Click Run button upper left with “iPhone 5.1 simulator” as the device.

3. Use the step and run trace buttons to step through the code after the system gets to a breakpoint. You can ‘float’ over variables to see their current values while tracing. The picture below shows a breakpoint on the log statement and two log results in the lower right (first and second rows displayed in a table).



Walkthrough the MainViewController.h and .m files and code. This is the view that lists all of the screens and allows you to tap and navigate to a screen. Note the logic to open new windows…this will be important. It is in MainViewController didSelectRowAtIndexPath method.

Also note the PageOneViewController add button logic in the viewDidLoad and addAction methods.

**PROBLEM SET #2**: Load and run the NavBar app in the simulator. Set breakpoints and run and step through the code when the simulator hits a breakpoint.

Try adding a NSLog statement in the addAction method and view the log after tapping the add button.

## User Interfaces

Good design guides from Apple:

<http://developer.apple.com/library/ios/#DOCUMENTATION/UserExperience/Conceptual/MobileHIG/Introduction/Introduction.html>

Really good elements guidelines document from Apple (must read):

[http://developer.apple.com/library/ios/#DOCUMENTATION/UserExperience/Conceptual/MobileHIG/UIElementGuidelines/UIElementGuidelines.html#//apple\_ref/doc/uid/TP40006556-CH13-SW1](http://developer.apple.com/library/ios/#DOCUMENTATION/UserExperience/Conceptual/MobileHIG/UIElementGuidelines/UIElementGuidelines.html)

### Navigation

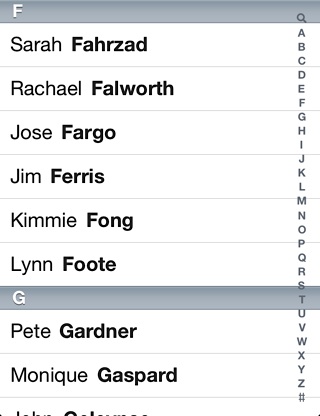
Two types of screen navigation are covered here and in storyboards which gives the ability to link together many screens. Storyboards are dwelt with in a later section.

#### Tab bar

The tabs at the bottom of an app can be set to display and navigate between 3+ screens quickly.

#### Navigation bar

\*Images from: *Apple UI Elements Guide*

UINavigationController

### Content Screen types

#### UIView

#### UIViewController

#### UITableView

#### UIWebView

#### UIPageViewController

#### UIAlertView and UIActionSheet





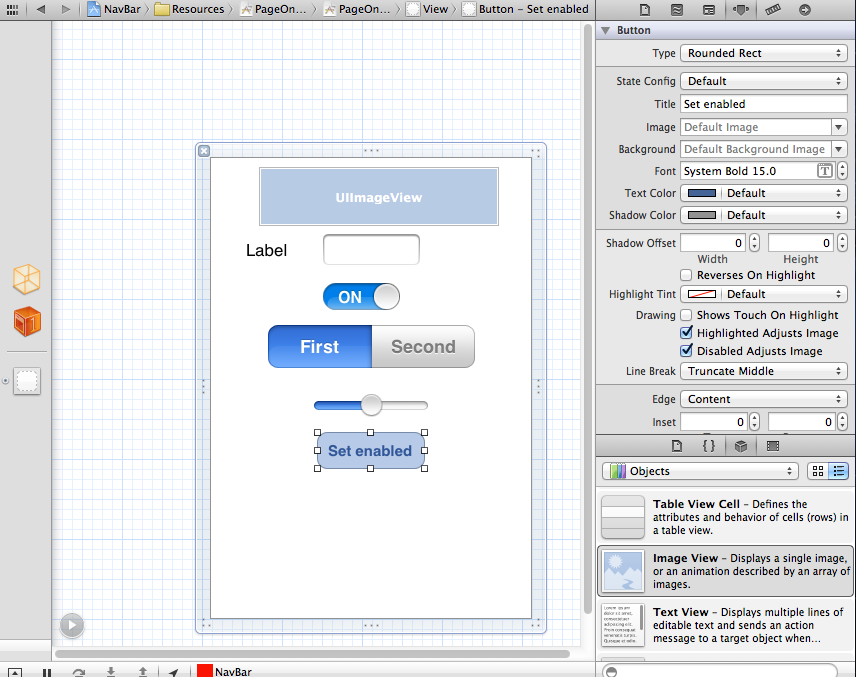
### Elements

We will now look at how to build an interface using user interface objects.

Go to Resources folder and click on the file (UI Screen): PageOneViewController.xib and make the image object smaller and add the controls from the library to the screen so that it looks like the image below. Add a label, text box, switch, segmented control, slider, and rounded rect button.

Note on the right, when you click on object, you get a **properties** viewer where you can change all sorts of default properties of the object such as its default text, color, alignment, font, etc. (To see the viewer you may need to click the 3rd icon in the View icons list to display the right hand window).

The properties are different for every object. Note that these properties can also be changed in code dynamically. So for instance, a button’s default text property can be set a text string like “Set Enabled” (do this when creating the button). Fonts and many other properties can also be changed. Explore the properties as you create the UI objects.



### Accessing elements in code

You will need to do two things with UI objects: **respond to events like the user tapping on them and access their object data.** Example: Respond when someone presses a button or read the text someone typed in a text field. Each UI element with corresponding code examples is listed below to do these two things: access data and/or respond to events.

In the example below we’re going to add UI elements to the first view: PageOneViewController. Make sure the PageOneViewController.h file looks like this:

#import <UIKit/UIKit.h>

@interface PageOneViewController : UIViewController {

UISwitch \*pwd\_on;

UIImageView \*myimage;

UILabel \*label1;

UITextField \*aName;

UIButton \*aButton;

UISegmentedControl \*mySeg;

UISlider \*mySlider;

int currMonth;

}

@property (nonatomic, retain) IBOutlet UISwitch \*pwd\_on;

@property (nonatomic, retain) IBOutlet UIImageView \*myimage;

@property (nonatomic, retain) IBOutlet UISlider \*mySlider;

@property (nonatomic, retain) IBOutlet UISegmentedControl \*mySeg;

@property (nonatomic, retain) IBOutlet UITextField \*aName;

@property (nonatomic, retain) IBOutlet UIButton \*aButton;

@property (nonatomic, retain) IBOutlet UILabel \*label1;

-(IBAction)setSlider:(id)sender;

-(IBAction)setEnabled:(id)sender;

-(IBAction)setSegment:(id)sender;

in .m right under the @implementation add the lines:

@synthesize pwd\_on, myimage, aName, aButton, label1, mySlider, mySeg;

-(IBAction)setSlider:(id)sender {

}

-(IBAction)setEnabled:(id)sender {

}

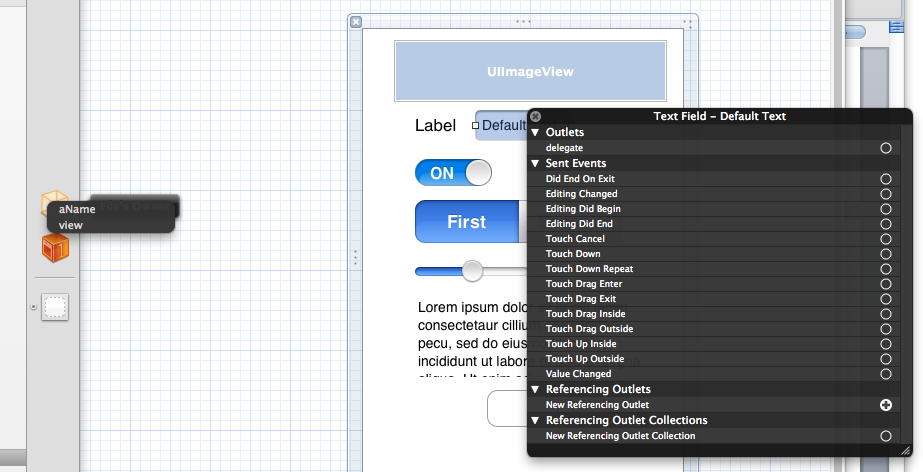
-(IBAction)setSegment:(id)sender {

}

To hook up the methods and variables to the screen, go to resources, PageOneViewController.xib. Right click on the control (like the text field as shown below) and drag from the circle next to “New Referencing Outlet” toward the bottom of the black box of events/outlets to the orange cube (File’s Owner is what the cube stands for which is a UIViewController object in this case) on the left and let the mouse go. A list of available variables will display as shown on the left below.

Select the variable to associate to the control…aName below. Set the label to label1 variable, textfield to aName, switch to pwd\_on, slider to mySlider, Segmented control (First, Second) to mySeg and the image to myimage.

To hook up the button click method drag circle *Touch up Inside* on button control to orange cube (File’s Owner) and select method: setEnabled. This means that when the user taps (not click’s, taps) the button the setEnabled code is executed automatically.



#### Labels

UILabel

It only makes sense to be able to set label text in code. Here is how to do that and how to set the font programmatically (note the font and other attributes can be set in the properties window within the IDE. Put this is the viewDidLoad method within the PageOneViewController.m file:

NSString \*sample=@"Test Text";

label1.text=sample;

label1.font = [UIFont fontWithName:@"Zapfino" size: 10.0];

#### UIImage

To display a new image from code (images can also be assigned in the properties viewer when the screen is created and never changed in the code).

in .m add the following to bottom of viewDidLoad method:

[self.title=@"Dale’s Screen](mailto:self.title=@%22Dale's%20Screen) ";

currMonth=2;

NSArray \*bigimage=[[NSArray alloc] initWithObjects:@"icon.png",@"email.png",nil];

myimage.image = [UIImage imageNamed:[bigimage objectAtIndex:currMonth-1]];

// zero based list so use currMonth-1 to get to the 2nd element

[bigimage release];

Note: you will need to add a email.png or other image to your Resources folder for this to work. If you change the name of the image make sure the code above has the new image filename.

Run the program and go to the first view. The image and label text should have changed. Now click in the text box…and try to get the keyboard to go away. It won’t…hmm.

#### Text input

UITextField and UITextView

To set a text value in code:

[aName.text=@"Sample](mailto:aName.text=@%22Sample) Text";

To read a text value the user has keyed in do this:

label1.text=aName.text;

This sets the label to the contents of the text field. We could assign the value in the text field to a string variable to use in our code:

NSString \*mystring;

mystring=aName.text;

#### Getting the keyboard to go away

1. Add <UITextFieldDelegate> in the PageOneViewController.h file so that the class definition appears as:

@interface PageOneViewController : UIViewController <UITextFieldDelegate>

2. Add to the viewDidLoad method at the bottom:

aName.delegate = self;

1. Add the method to the .m file at the bottom (above @end).

- (BOOL)textFieldShouldReturn:(UITextField \*)textField

{

    [aName resignFirstResponder];

label1.text=aName.text;

currMonth=1;

    return YES;

}

Now run the program, type in the text box and hit “Return” on the keyboard and the keyboard should go away AND the text you typed will be put in the label.

### Selection Elements

The following elements require code to respond to a tap action by the user.

#### UIButton

Lets put the following setHidden function for the label in the button method. That way, when the button is tapped the label will toggle on and off. (The ! symbol is the Boolean NOT and makes current value of false=true and vice versa)

-(IBAction)setEnabled:(id)sender {

[label1 setHidden:(!label1.hidden)];

}

Run and tap the button a few times. The Label should appear and disappear from the screen.

If the button isn’t responding, did you set the “New Referencing Outlet” to aButton within the .xib for the screen? This connects the button up so it is recognized.

#### UISwitch

A switch allows a Boolean value to be set. To retrieve the value use code such as:

BOOL switchValue=pwd\_on.isOn;

Add the code to the textFieldShouldReturn method at the bottom of it before the “return YES;” statement. Make sure the label is big enough in the .xib file to display the text and the “On:” string.

BOOL switchValue=pwd\_on.isOn;

if (switchValue) {

label1.text=[aName.text stringByAppendingString:@"On: "];

}

#### UISegmentedControl

Through the properties you can add one or more segments to the segment control. Add this code to the bottom of the viewDidLoad routine and run the app.

mySeg.segmentedControlStyle = UISegmentedControlStyleBar;

mySeg.selectedSegmentIndex = 1; //zero based: 0 is first segment

This initializes the value of the segement control to the second position.

To see the value the user set, query the value of selectedSegmentIndex. This code in setSegment sets the label depending on which segment is selected. What must be done with setSegment method before it will work? (**Answer: it must be associated with the segment control in the UI tool using the “value changed” event)**

-(IBAction)setSegment:(id)sender {

if (mySeg.selectedSegmentIndex==0) {

[label1 setText:@"Segment 1"];

}

else {

[label1 setText:@"Segment 2"];

}

}

#### UISlider

The slider has a float numeric value. From the properties screen for the UISlider you can set a min, max and default value for the control. Set ours to min=0 and max=100. Then use the following code in the setSlider function (hook up setSlider to the “value changed” event in the UI builder tool). The code will display the slider value in the text field.

-(IBAction)setSlider:(id)sender {

aName.text=[NSString stringWithFormat:@"Slider: %3.0f",mySlider.value];

}

#### UIPickerView, UIDatePicker

These controls allow the user to select from a list of known values or select a date. See Cookbook or Google these class names for example of how to use picker and date picker controls.

#### Controlling view methods

There are several standard methods that get called whenever a view (screen) is navigated to (opened) or navigated away from (closed). These methods are often commented out in a new view .m file and you can just uncomment them and use them.

The most common is the viewDidLoad method. Put most initialization logic in here. Example:

- (void)viewDidLoad {

self.title=@"Dale’s Screen "; /\* set the title of the screen when it opens \*/

UIBarButtonItem \*mailButton = [[UIBarButtonItem alloc] initWithImage:[UIImage imageNamed:@"icon\_mail.png"] style:UIBarButtonItemStylePlain target:self action:@selector(emailPDF:)];

self.navigationItem.rightBarButtonItem = mailButton;

[mailButton release];

self.view.backgroundColor = [UIColor clearColor];

[super viewDidLoad];

// Uncomment the following line to display an Edit button in the navigation bar for this view controller.

// self.navigationItem.rightBarButtonItem = self.editButtonItem;

}

viewWillAppear gets called right before the view appears. Note that viewDidLoad is called after the graphics are displayed so viewWillAppear code gets called before this, before graphic images are displayed. viewWillDisappear and viewDidUnload are available when the view is navigated away from

PROBLEM SET #3: Apply all user interface changes above. Show your working app in the simulator with sample values in the interface elements that show the functionality created.

**Create your own screen in the PageFourViewController with at least three UI elements that have logic behind them.**

## Objects and Objective C

IOS 5 Programming Cookbook pages 21-83 has a good summary and examples of the Objective C programming language.

### Objects

Representations of real-world (or at least programmatic) things *(nouns)* and define the attributes, *properties*, and behavior, *methods*, of those ‘things’. A screen, text box, app itself are objects. Objects ‘link’ or ‘call’ one another.

A program is really many objects that encapsulate information and respond to one another or ‘be called’ to perform various actions when needed. When an action, or *event*, happens (like the user touched a button or a screen was opened) objects can be programmed to respond to these events.

Essentially each user interface element or data construct is an object. Objects are called Classes. Example: the class UIButton is a button object class (see pageoneviewcontroller.h)

Encapsulation - Really important…its what makes an object useful and self-contained entity that can be defined and *instantiated* many times…each copy keeping its own data.

Inheritance - Subclasses allow a hierarchy of object classes with *inherited* pre-defined behavior and attributes from parent classes.

***Note that a LOT of good examples can be found by using Google or search engines with terms such as iOS and the object name…example “setting iOS NSString”***

Note: Comments in code are /\* comment here \*/ for blocks or // before a line to comment just the line.

***Example User interface object***

Click on MainViewController.h which is a class header for class MainViewController object which is a subclass of class UIViewController. Notice the import says we need the header for the the ModalViewController screen type as we access that object type. UIKit defines the classes for general UI screens like the UIViewController.

Then each property or method we want to expose externally is defined. There properties: myTableView (we’ll talk about the UITableView later), an array data object: menuList which is an object of class: NSMutableArray, and myModalViewController which is of object type ModalViewController. Note the \* means it is an object.

#import <UIKit/UIKit.h>

#import "ModalViewController.h"

@interface MainViewController : UIViewController <UINavigationBarDelegate, UITableViewDelegate, UITableViewDataSource, UIActionSheetDelegate>

{

UITableView \*myTableView;

NSMutableArray \*menuList;

ModalViewController \*myModalViewController;

}

@property (nonatomic, retain) IBOutlet UITableView \*myTableView;

@property (nonatomic, retain) NSMutableArray \*menuList;

@property (nonatomic, retain) ModalViewController \*myModalViewController;

(IBAction)styleAction:(id)sender;

(void) setRow :(int)arow :(NSString \*)astring;

(int) returnRow :(NSString \*)searchString;

@end

### Data

Properties contain variables and sets of data in arrays (which are data objects). The NSMutableArray object above is an array of objects. Mutable means it is not static and we can add or remove elements of the list or array once it has been created. This object has built-in methods like addObject and removeObject that allows us to manipulate array elements.

Many data and user interface are defined in the libraries of code that come with the iOS development framework.If you go to Help/API Reference you can search for the NSMutableArray class reference help and it will define all available methods for the object. All objects are in the API Reference so you can find the right actions or methods to use when needed for the objects.

### Initializing

***Examples:***

To define the data object so it can be accessed externally from the object, in the .h header you put:

@interface MainViewController : UIViewController

{

NSMutableArray \*menuList;

int val1;

NSString \*astring;

float val2;

BOOL val3;

}

@property (nonatomic, retain) NSMutableArray \*menuList;

@property (nonatomic, retain) NSString \*astring;

@end

and in the .m you need to add the variable to the @synthesize command:

@synthesize menuList, astring;

You don’t need to add the int, float, BOOL data types are not objects and so don’t need property and synthesize statements.

To just add a static variable that is only available within methods of the object (not externally) use a definition such as the following at the top of the .m file:

static NSArray \*pageNames = nil;

To manipulate data in an object variable, commands like the following can be used.

**To initialize values in the array:**

self.menuList = [NSMutableArray array];

Note the “self.” code means access a property in the current (self) object. The period is used between the object and a property or method name in Objective C (or in C++).

**To add objects to the array:**

pageNames = [[NSArray alloc] initWithObjects:@"PageOne", @"PageTwo", @"PageThree", @"PageFour", @"PageFive", nil];

Note that this initializes the array with five string objects. String objects use @”value” syntax to define a string.

Below a NSMutatbleDictionary object is added to the menuList array. Each of these objects in turn contains two elements: Title and Explain. A Dictionary type object takes pairs of strings: a value and a key.

NSMutableArray \*patientData;

if (patientData==nil)

patientData = [NSMutableArray new];  
else {

[patientData removeAllObjects];

}

NSDictionary \*row1=[[NSDictionary alloc] initWithObjectsAndKeys:[NSNumber numberWithInt:100], @"patient\_id", [NSNumber numberWithInt:1], @"priority",@”Patient 1”, @"patient\_name",[NSNumber numberWithInt:0], @"min\_type", nil];

**[patientData addObject:row1];**

[row1 release];

To access an element in the dictionary:

int row=1;

NSDictionary \*rowdata = [patientData objectAtIndex:row];

int patient\_id=[[rowdata objectForKey:@"patient\_id"] intValue];

### Methods

The processes or actions that an object can do. Many methods and attributes are pre-developed for user interface and data object classes and 80% of our code will be placed within these pre-defined methods.

Other methods we will create ourselves. Example methods we create:

(IBAction)styleAction:(id)sender;

(void) setRow :(int)arow :(NSString \*)astring;

(int) returnRow :(NSString \*)searchString;

Return value is within the ( ). Parameters passed to the function method are after the : symbols. Note :(int)arow means the integer parameter arow is passed into the method. To call the method we might have:

NSString \*test=@”Test Row”;

Int ret=[self returnRow:test];

This passes variable test to the function returnRow which is in the current object (self). Ret receives the integer result.

Note the syntax to reference a method…just memorize it: [objectname methodname:parameter1 :parameter\_i]; We could have as many parameters after : symbols as needed, or one or no parameters. So to call setRow we might have: [objectname\_var setRow:5 :@”Test”];

### Code: processing logic inside a method

Code is about calculations and data state changes and uses three constructs: assignment, conditional branching, and iteration. You’ve already seen a number of these with the = symbol above. Calculations might be a=5\*b; or to define a new variable and initialize it using a calculation you might write: int d=a\*2;

Note the symbol ; ends a complete statement.

**Conditional Branching**: testing values and branching

if (!pageNames)

{

pageNames = [[NSArray alloc] initWithObjects:@"PageOne", @"PageTwo", @"PageThree", @"PageFour", @"PageFive", nil];

}

! means not and == means equal in a condition statement. && means and and || means or as in other examples:

if ((a==b) && (c<2)) {

…statements…

}

else {

…statements…

}

Where the statements within the { and } are executed if the condition is true (or it is false and executes statements for the else portion of the condition.

String testing:

if (!(pwd\_on.isOn) || ([aRate.text isEqualToString:aemail.text])) {

… }

**Iteration:** looping and repeating a process for an entire set of data:

for (NSString \*pageName in pageNames)

{

…statements…

}

Note that statements can access the pageName object/variable which changes with each iteration. Other examples of the for and while loop constructs see pages cookbook pages 31-35.

while (a < b) {

a=a+1; // or a+=1;

}

**CODING PROBLEM SET #4**

a) Add logic (code) to NavBar app screen:PageTwoViewController to test for user tap on a new user interface button named “Test” and if the value in a text box is >1,000,000 put an alert message up. You will need to add a button and text box and then the logic.

b) After this works, add the logic to create a (void) method (one that doesn’t return a value) called InputCheck that you call from the button TouchUpInside event. Pass as parameter: text box contents as a string value. The method should convert the string to an integer within its logic (use: NSStringname.intValue).

**OR** if you are more advanced at programming add multiple buttons and create a calculator that tests for illegal input characters (your InputCheck method should ensure no characters or illegal punctuation exists in the input).

*Alert view sample code which is needed:*

UIAlertView \*alert=nil;

UIAlertView \*alert=[[UIAlertView alloc] initWithTitle: @"Information Needed" message:@"Please fill in your name, email and daily rate to set up the application. Try again?" delegate:self cancelButtonTitle:@"No" otherButtonTitles:@"Yes", nil];

[alert show];

[alert release];

- (void)alertView:(UIAlertView \*)alertView willDismissWithButtonIndex:(NSInteger)buttonIndex

{

if (buttonIndex == 1) {

//logic for tab of ‘Yes’ response

}

}

## 5. Table views

All lists of information are displayed through a UITableView object which will be used in about 75% of your screens that you build. The following techniques display data in table view cells and allow selection actions to occur on the cells. Note that cells themselves may be added, deleted, moved, and headers, footers added to the table view. Also, table view cells can contain customizable ‘screens’ … multiple UI elements such as text boxes, buttons, labeles, etc. within *each* cell.

### Adding a table to UIViewController

#### .h changes

Make the PageTwoViewController.h appear as the following. Note the use of the UITableview delegate/datasource statements in the interface.

#import <UIKit/UIKit.h>

@interface PageTwoViewController : UIViewController <UITableViewDelegate, UITableViewDataSource, UIActionSheetDelegate>

{

UITableView \*myTableView;

}

@property (nonatomic, retain) IBOutlet UITableView \*myTableView;

@end

#### .m changes

Add the following to PageTwoViewController.m

@synthesize myTableView;

#pragma mark -

#pragma mark UITableViewDataSource

- (NSInteger)tableView:(UITableView \*)tableView numberOfRowsInSection:(NSInteger)section

{

return 1;

}

- (UITableViewCell \*)tableView:(UITableView \*)tableView cellForRowAtIndexPath:(NSIndexPath \*)indexPath

{

static NSString \*kCellIdentifier = @"cellID";

UITableViewCell \*cell = [tableView dequeueReusableCellWithIdentifier:kCellIdentifier];

if (!cell)

{

cell = [[[UITableViewCell alloc] initWithStyle:UITableViewCellStyleValue1 reuseIdentifier:kCellIdentifier] autorelease];

cell.accessoryType = UITableViewCellAccessoryDisclosureIndicator;

cell.textLabel.backgroundColor = [UIColor clearColor];

cell.textLabel.opaque = NO;

cell.textLabel.textColor = [UIColor blackColor];

cell.textLabel.highlightedTextColor = [UIColor whiteColor];

cell.textLabel.font = [UIFont boldSystemFontOfSize:18];

cell.detailTextLabel.backgroundColor = [UIColor clearColor];

cell.detailTextLabel.opaque = NO;

cell.detailTextLabel.textColor = [UIColor grayColor];

cell.detailTextLabel.highlightedTextColor = [UIColor whiteColor];

cell.detailTextLabel.font = [UIFont systemFontOfSize:24];

}

cell.textLabel.text =@"Test";

return cell;

}

#pragma mark -

#pragma mark UITableViewDelegate

- (void)tableView:(UITableView \*)tableView didSelectRowAtIndexPath:(NSIndexPath \*)indexPath

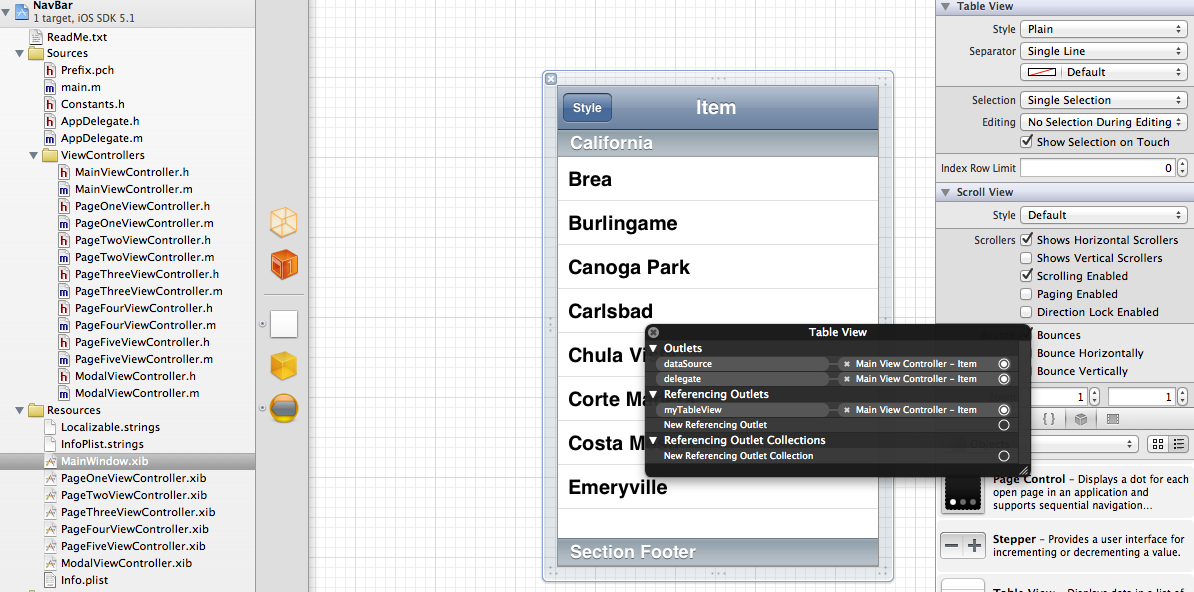
{

#### }

#### .xib changes

Add a UITableView to the UIView for the pagetwoviewcontroller.xib file. Right-click on it and drag the

Datasource and delegate outlets to fileOwner (The orange 3-d box).



The tableView class, cellForRowAtIndexPath method displays cell data. It is called once for each cell in the tableView. The number of cells to be displayed is defined by the numberOfRowsInSection method.

Run in the simulator.

### Adding an array to hold data to be displayed

.h

#import <UIKit/UIKit.h>

@interface PageTwoViewController : UIViewController <UITableViewDelegate, UITableViewDataSource, UIActionSheetDelegate>

{

UITableView \*myTableView;

NSMutableArray \*listData;

}

@property (nonatomic, retain) IBOutlet UITableView \*myTableView;

@property (nonatomic, retain) NSMutableArray \*listData;

@end

.m

Change synthesize statement at top of .m to:

@synthesize myTableView, listData;

- (void)viewDidLoad

{

// add our custom image button as the nav bar's custom right view

UIBarButtonItem \*addButton = [[UIBarButtonItem alloc] initWithImage:[UIImage imageNamed:@"email.png"]

style:UIBarButtonItemStyleBordered target:self action:@selector(action:)];

self.navigationItem.rightBarButtonItem = addButton;

[addButton release];

if (listData==nil)

listData = [NSMutableArray new];

else {

[listData removeAllObjects];

}

int name\_id=1;

NSString \*aname=@"Name 1";

NSDictionary \*row1=[[NSDictionary alloc] initWithObjectsAndKeys:[NSNumber numberWithInt:name\_id], @"id", aname, @"name",nil];

[listData addObject:row1];

[row1 release];

name\_id=2;

aname=@"Name 2";

NSDictionary \*row2=[[NSDictionary alloc] initWithObjectsAndKeys:[NSNumber numberWithInt:name\_id], @"id", aname, @"name",nil];

[listData addObject:row2];

[row2 release];

}

last rows of cellForRowAtIndexPath method changed to:

NSUInteger row = [indexPath row];

NSDictionary \*rowdata = [listData objectAtIndex:row];

cell.textLabel.text = [rowdata valueForKey:@"name"];

cell.detailTextLabel.text = [NSString stringWithFormat:@"id: %d",[[rowdata valueForKey:@"id"] intValue]];

NSLog (@"Table text:%@ rows #:%d", [rowdata valueForKey:@"name"], indexPath.row);

return cell;

Change numberofrowsinsection method to:

- (NSInteger)tableView:(UITableView \*)tableView numberOfRowsInSection:(NSInteger)section

{

return [self.patientData count];

}

Run in simulator and should get two rows in list.

Note in the code above textLabel and detailTextLabel are standard default objects of the cell object. Custom cells are defined below.

### Responding to cell taps

Add the following to the top of PageTwoViewController.m

#import "PageFourViewController.h"

#import "PageFiveViewController.h"

Add at the end of the .m file:

-(void)tableView:(UITableView \*)tableView didSelectRowAtIndexPath:(NSIndexPath \*)indexPath {

NSUInteger row = [indexPath row];

if (row<[self.listData count]) {

PageFourViewController \*fourview = [[PageFourViewController alloc] initWithNibName:@"PageFourViewController" bundle:nil];

fourview.name=@"Tests";

[self.navigationController pushViewController:fourview animated:YES];

[fourview release];

}

}

The following code changes are only needed because we set the name variable up as an illustration of passing data to the view from the calling parent view. Make PageFourViewController.h appear as:

#import <UIKit/UIKit.h>

@interface PageFourViewController : UIViewController

{

UIColor \*defaultTintColor;

NSString \*name;

}

@property (nonatomic, retain) NSString \*name;

@end

Make PageFourViewController.m appear as:

@synthesize name;

- (void)viewDidLoad

{

self.title=name;

…

### Responding to cell ‘arrow’ tap

Include the following line in the cellForRowAtIndexPath method for a cell. It illustrates bringing up an alertView question dialog if the user taps the blue > arrow next to a cell.

cell.accessoryType = UITableViewCellAccessoryDetailDisclosureButton;

- (void)tableView:(UITableView \*)tableView accessoryButtonTappedForRowWithIndexPath:(NSIndexPath \*)indexPath {

UIAlertView \*alert=[[UIAlertView alloc] initWithTitle: @"Information Needed" message:@"Please fill in your name, email and daily rate to set up the application. Try again?" delegate:self cancelButtonTitle:@"No" otherButtonTitles:@"Yes", nil];

[alert show];

[alert release];

}

- (void)alertView:(UIAlertView \*)alertView willDismissWithButtonIndex:(NSInteger)buttonIndex

{

if (buttonIndex == 1) {

//logic for tab of ‘Yes’ response

PageFiveViewController \*fiveview = [[PageFiveViewController alloc] initWithNibName:@"PageFiveViewController" bundle:nil];

[self.navigationController pushViewController:fiveview animated:NO];

[fiveview release];

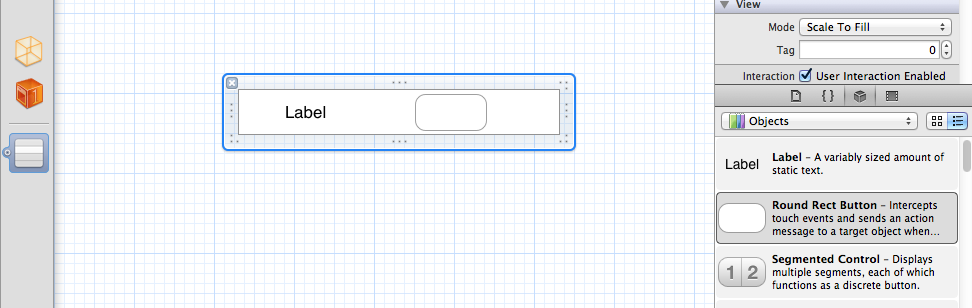
}

}

### Creating a custom cell

Select Resources folder on left in XCODE environment. Go to File>New>File and select on the left UI and then View on the right and name it “Custom1”.

An xib is created. Delete the view that is shown then drag/drop a table cell object into the drawing area from the UI tools palette. Add a label and button controls to the view. You should have:



Using properties screen on right click on label and set its ‘Tag’ to 100 and the tag for the button to 101.

In PageTwoViewController.m replace the cellForRowAtIndexPath method with the code below. The bolded code enables the custom cell for the first cell (only) in the list:

- (UITableViewCell \*)tableView:(UITableView \*)tableView cellForRowAtIndexPath:(NSIndexPath \*)indexPath

{

UITableViewCell \*cell;

**if (indexPath.row==0) {**

**cell = [tableView dequeueReusableCellWithIdentifier:@"Custom1"];**

**if (!cell)**

**cell = [[[NSBundle mainBundle] loadNibNamed:@"Custom1" owner:self options:nil] lastObject];**

**[(UILabel \*)[cell viewWithTag:100] setText:@"Testing Label"];**

**}**

else {

static NSString \*kCellIdentifier = @"cellID";

cell = [tableView dequeueReusableCellWithIdentifier:kCellIdentifier];

cell = [[[UITableViewCell alloc] initWithStyle:UITableViewCellStyleValue1 reuseIdentifier:kCellIdentifier] autorelease];

cell.accessoryType = UITableViewCellAccessoryDetailDisclosureButton;

cell.textLabel.backgroundColor = [UIColor clearColor];

cell.textLabel.opaque = NO;

cell.textLabel.textColor = [UIColor blackColor];

cell.textLabel.highlightedTextColor = [UIColor whiteColor];

cell.textLabel.font = [UIFont boldSystemFontOfSize:18];

cell.detailTextLabel.backgroundColor = [UIColor clearColor];

cell.detailTextLabel.opaque = NO;

cell.detailTextLabel.textColor = [UIColor grayColor];

cell.detailTextLabel.highlightedTextColor = [UIColor whiteColor];

cell.detailTextLabel.font = [UIFont systemFontOfSize:14];

// get the view controller's info dictionary based on the indexPath's row

NSUInteger row = [indexPath row];

NSDictionary \*rowdata = [listData objectAtIndex:row];

cell.textLabel.text = [rowdata valueForKey:@"name"];

cell.detailTextLabel.text = [NSString stringWithFormat:@"id: %d",[[rowdata valueForKey:@"id"] intValue]];

NSLog (@"Table text:%@ rows #:%d", [rowdata valueForKey:@"name"], indexPath.row);

}

return cell;

}

Run in simulator and go to view 2 and you’ll see the custom code

### Delete or add cells

Cookbook page 268

### Headers/footers

See cookbook page 270 section 3.9.

Problem Set #5: Create a new app!

A) using the File new project function. Select a tabbar app type. The app should have two screens. The first will be a table (list) view listing a set of objects – classmate names or whatever you want. Store the data in an NSMutatbleDictionary. Save the data with key and value pairs. A lot of the code will be similar to the NavBar TableView code and can be copied from there.

B) Add code to display an actionsheet of choices when a cell is selected. See code within MainViewController for action sheets and note that in the .h use must add the UIActionSheetDelegate text to the interface definition of the object name. When the first choice is selected in the actionsheet bring up a WEBUI screen that displays a web page related to the object selected.

## 6. Application Data and Storage

### iPhone Settings

Settings for each app can be visible within the device settings screen if you use a settings.bundle root.plist file within resources. This image shows an example of this file. All settings are configured (no code involved).

To access a setting in code use this sample as an example:

NSUserDefaults \*defaults =[NSUserDefaults standardUserDefaults];

int allow\_email\_billing=[[defaults objectForKey:kallow\_email\_billing] intValue];

where kallow\_email\_billing is a constant string value such as:

#define kallow\_email\_billing @"allow\_email"

This is defined within a .h file after the #import statements.

Defaults may also be set…see Apple documentation by searching for iOS NSuserdefaults.

### Files

Images are provided as resources. Other text or PDF documents can be included as resources as well or created in the app itself…yes, PDF documents can be created but it takes another library of routines to do it.

### Local Database

SQLite is the local device database of choice and can store a lot of local data and is easily updatable. It makes the apps useful without having to be connected to external servers and is great for storing user data OR for storing pre-set up data when the app ships. However, this pre-setup data should not change often as an update to the app is required to update the data. Still, it is a powerful way of storing and utilizing data for the app.

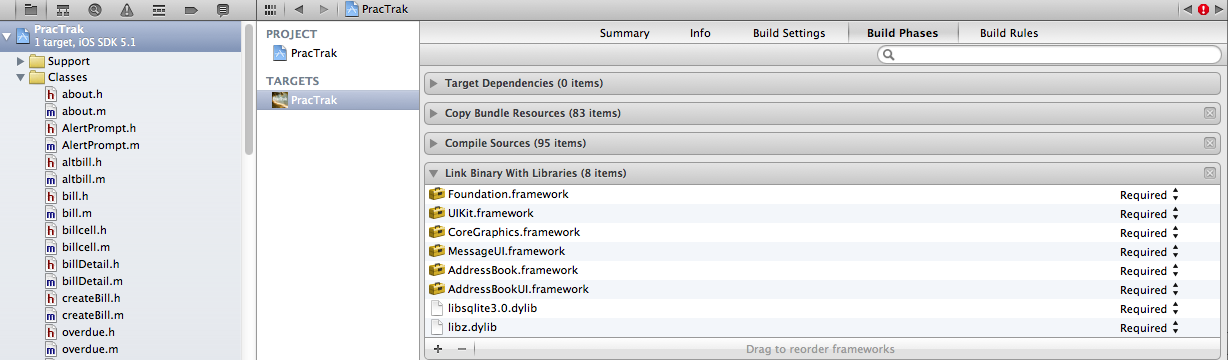
Code to set up app to use sqlite3.

Frameworks need to be added to the app:

1. libsqlite3.0.dylib

2. libz.dylib

To do this click on the main app name (shown highlighted on the left below) then click “Build Phases” and open “Link Binary with Libraries” and press the + button and add libraries above.



The following code isn’t part of the exercise but are required changes to each .h and .m code file that will be accessing data from sqlite3.

### .h changes:

#import "sqlite3.h"

#define kFilename @"data.sqlite3"

@interface about : UIViewController {

sqlite3 \*database;

}

(NSString \*) dataFilePath;

@end

### .m changes:

- (NSString \*)dataFilePath {

NSArray \*paths = NSSearchPathForDirectoriesInDomains(NSDocumentDirectory, NSUserDomainMask, YES);

NSString \*documentsDirectory = [paths objectAtIndex:0];

return [documentsDirectory stringByAppendingPathComponent:kFilename];

}

### Create the database if it isn’t already created

Create two files: utils.h and utils.m with the content:

utils.h:

#import <UIKit/UIKit.h>

#import "sqlite3.h"

#define kFilename @"data.sqlite3"

@interface utils : NSObject {

sqlite3 \*database;

}

-(NSString \*)dataFilePath;

-(void) checkDB;

@end

utils.m:

#import "utils.h"

@implementation utils

- (NSString \*)dataFilePath {

NSArray \*paths = NSSearchPathForDirectoriesInDomains(NSDocumentDirectory, NSUserDomainMask, YES);

NSString \*documentsDirectory = [paths objectAtIndex:0];

return [documentsDirectory stringByAppendingPathComponent:kFilename];

}

-(void) checkDB {

NSLog(@"Database path:%@",[self dataFilePath]);

if (sqlite3\_open([[self dataFilePath] UTF8String], &database) != SQLITE\_OK) {

sqlite3\_close(database);

NSAssert(0, @"Failed to open database");

}

char \*errorMsg;

NSString \*createSQL = @"CREATE TABLE IF NOT EXISTS CUST (CUST\_ID INTEGER, NAME TEXT, DATETIMESTAMP TEXT, SKU INTEGER, quantity REAL, Description TEXT);";

if (sqlite3\_exec (database, [createSQL UTF8String],NULL, NULL, &errorMsg) != SQLITE\_OK) {

sqlite3\_close(database);

NSAssert1(0, @"Error creating table: %s", errorMsg);

}

sqlite3\_close(database);

}

@end

Call the checkDB to create the database when the app opens. Add the following code to `.m:

#import "utils.h"

At the end of applicationDidFinishLaunching:

utils \*autil = [[utils alloc] init];

[autil checkDB];

[autil release];

The new database is stored in a path such as the following…this path will appear in the log when the application starts.

**/Users/dale.matheny/Library/A pplication Support/iPhone Simulator/5.1/Applications/9A16D808-803A-412D-B008-B6ABFF62A761/Documents/**

Open the database using your sqlite tool which can be downloaded for free here:

<http://downloads.yahoo.com/software/macintosh-office-productivity-navicat-lite-free-multiple-databases-gui-for-mac-os-x--s372725>

Add rows into the database for testing.

To load data into the table we add a loadData routine as follows. Add the database .h stuff to support databases (see above). Then add into the 2nd page view .m file:

At bottom of viewDidLoad (replacing any existing initialization of the listData variable):

[self loadData];

Add the method then run the app and go to the list on page 2. If you’ve entered data in the CUST table in the database then it should appear.

-(void)loadData {

//save password to database

if (sqlite3\_open([[self dataFilePath] UTF8String], &database) != SQLITE\_OK) {

sqlite3\_close(database);

NSAssert(0, @"Failed to open database");

}

int min\_id=100;

int cust\_id=0;

NSString \*select = @"SELECT cust\_id,name from cust where cust\_id>?;";

sqlite3\_stmt \*statement;

if (sqlite3\_prepare\_v2( database, [select UTF8String],-1, &statement, nil) == SQLITE\_OK) {

sqlite3\_bind\_int(statement, 1, min\_id);

while (sqlite3\_step(statement) == SQLITE\_ROW) {

cust\_id=sqlite3\_column\_int(statement,0);

char \*namec = (char \*)sqlite3\_column\_text(statement, 1);

if (namec==nil) namec="";

NSString \*name = [NSString stringWithUTF8String:namec];

NSDictionary \*row1=[[NSDictionary alloc] initWithObjectsAndKeys:[NSNumber numberWithInt:cust\_id], @"id", name, @"name",nil];

[listData addObject:row1];

[row1 release];

}

sqlite3\_finalize(statement);

}

sqlite3\_close(database);

}

#### Example of an Insert, update, and delete of data in a database

char \*update = "begin;";

if (sqlite3\_exec (database,update,NULL, NULL, &errorMsg) != SQLITE\_OK) {NSLog(@"Error creating table: %s", errorMsg);}

sqlite3\_stmt \*stmt;

char \*insert = "INSERT INTO PREFS (patient\_id,pref,value) values (?,'EMAIL\_BILL',?);";

if (sqlite3\_prepare\_v2(database, insert , -1, &stmt, nil) == SQLITE\_OK) {

sqlite3\_bind\_int(stmt, 1, patient\_id);

sqlite3\_bind\_text(stmt, 2, [avalue UTF8String], -1, NULL);

}

if (sqlite3\_step(stmt) != SQLITE\_DONE) NSAssert(0, @"Error updating table");

sqlite3\_finalize(stmt);

update = "UPDATE BILLS set DISCOUNT=?, disc\_type=?, TOTAL=? WHERE BILL\_ID=?;"; //rate=?,

if (sqlite3\_prepare\_v2(database, update, -1, &stmt, nil) == SQLITE\_OK) {

sqlite3\_bind\_double(stmt, 1, discount);

sqlite3\_bind\_int(stmt, 2, disc\_type);

sqlite3\_bind\_double(stmt, 3, total);

sqlite3\_bind\_int(stmt, 4, bill\_id);

}

if (sqlite3\_step(stmt) != SQLITE\_DONE) NSLog(@"Error updating table in bill detail");

sqlite3\_finalize(stmt);

update = "DELETE FROM NOTES WHERE BILL\_ID=? AND TYPE=4;";

if (sqlite3\_prepare\_v2(database, update, -1, &stmt, nil) == SQLITE\_OK) {

sqlite3\_bind\_int(stmt, 1, bill\_id);

}

if (sqlite3\_step(stmt) != SQLITE\_DONE)

NSAssert(0, @"Error updating table");

sqlite3\_finalize(stmt);

//remove \n if at end of notes

NSString \*note=self.patient\_bill\_notes;

if ([note length]>=1) {

int fromint=[note length]-1;

NSString \*aend=[note substringFromIndex:fromint];

if ([aend isEqualToString:@"\n"])

note=[note substringToIndex:fromint];

}

NSDateFormatter \*dateFormat = [[NSDateFormatter alloc] init];

[dateFormat setDateFormat:@"yyyy-MM-dd HH:mm:ss +0000"];

NSDate \*today = [[NSDate alloc] init];

NSString \*timenow = [dateFormat stringFromDate:today];

[today release];

[dateFormat release];

NSAssert(0, @"Error updating table");

sqlite3\_finalize(stmt);

sqlite3\_close(database);

Problem Set #6: for the new app created in problem set #5 add a database to load the data into the table.

Bonus: add insert, update, delete functionality to the list. See iOS5 Programming cookbook for how to add and delete list elements.

# 7. Adding a UIwebView

### Within view 4 .h:

#import <UIKit/UIKit.h>

@interface PageFourViewController : UIViewController

{

UIColor \*defaultTintColor;

NSString \*name;

IBOutlet UIWebView \*webView;

}

@property (nonatomic, retain) UIWebView \*webView;

@property (nonatomic, retain) NSString \*name;

@end

### Within the pageFourViewController.xib

Add a web view (delete the image that is there). Right click and hook up the referencing outlet to the file owner (3-d box) object webView object.

**Within view 4 .m:**

@synthesize name,webView;

within viewDidLoad at bottom:

NSString \*urlAddress=[NSString stringWithFormat:@"%@",@"http://www.google.com"];

//Create a URL object.

NSURL \*url = [NSURL URLWithString:urlAddress];

//URL Requst Object

NSURLRequest \*requestObj = [NSURLRequest requestWithURL:url];

//Load the request in the UIWebView.

[webView loadRequest:requestObj];

Add the framework: mapkit.framework

To view 5 .h:

#import <UIKit/UIKit.h>

#import <MapKit/MapKit.h>

#define METERS\_PER\_MILE 1609.344

@interface PageFiveViewController : UIViewController

{

UIColor \*defaultTintColor;

MKMapView \*mapView;

}

@property (nonatomic, retain) IBOutlet MKMapView \*mapView;

@end

Within the .m for view 5:

@synthesize mapView;

At bottom of viewWillAppear:

self.mapView.mapType = MKMapTypeSatellite;

CLLocationCoordinate2D zoomLocation;

zoomLocation.latitude = 38.9490;

zoomLocation.longitude= -90.3475;

MKCoordinateRegion viewRegion = MKCoordinateRegionMakeWithDistance(zoomLocation, .5\*METERS\_PER\_MILE, .5\*METERS\_PER\_MILE);

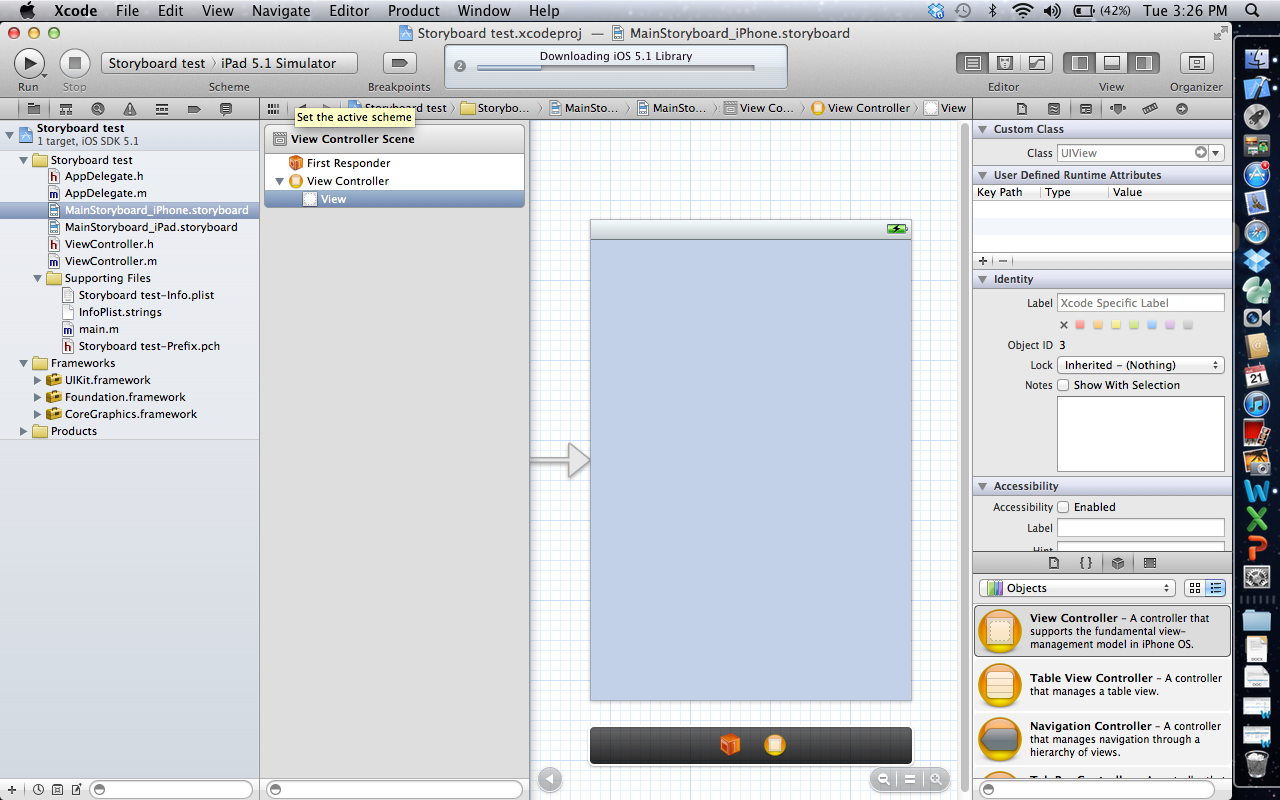
MKCoordinateRegion adjustedRegion = [mapView regionThatFits:viewRegion];

[mapView setRegion:adjustedRegion animated:YES];

Run the app.

## 7. Navigation using storyboards

See iOS 5 programming cookbook chapter 4 page 305-320.



Problem set #7: follow the instructions in the IOS 5 cookbook chapter 4 to develop a simple storyboard app.

## 8. Integrating with the world

The real power of mobile apps comes with accessing cloud (web) server processes or *web services*. This allows massive data and processing to be provided to small mobile devices…lots of power and data available *anywhere*.

* RSS (Really simple syndication) Feeds – Standardized tag based content. Very consistent and easy to read. Many Website tools can be set to generate content into RSS feeds automatically. Accessed via URL.
* Screen scraping – pulling data from HTML screens by searching for tags and related content
* Web services -- can be easily created with PHP scripts on external servers that access or update databases.

### RSS Feeds

* Example for CNN: <http://www.cnn.com/services/rss/>

For top stories use URL: feed://rss.cnn.com/rss/cnn\_topstories.rss

Source example:

<div class="apple-rss-article apple-rss-read" onclick="handleArticleClick(this)" showSeparator="true"

articlesortdate="0367710961.000000" articlesorttitle="hiker photographs grizzly before dying" articlesortsource="" sourceindex="0" articlesortid="00000000000000000115" articlelocaldate="0367711272.343473" articleid="799302648bfc6c07d25e9cc0e76806ec0a143fcd">

<div class="apple-rss-article-footer"></div>

<div class="apple-rss-article-head" >

<div class="apple-rss-unread-dot"><img src="feed://\_\_rsrc\_\_/\_\_rsrc\_\_/unread.tif" width="9" height="9" /></div>

<div class="apple-rss-subject" title="Hiker photographs grizzly before dying"><a href="http://rss.cnn.com/~r/rss/cnn\_topstories/~3/9v85ETpzP68/index.html">Hiker photographs grizzly before dying</a></div>

<div class="apple-rss-summary" >A southern California man killed by a grizzly bear in Alaska&apos;s backcountry was shooting photos of the animal that killed him just moments before the attack, a National Park Service official said Sunday.</div>

<div class="apple-rss-date" title="Today, 4:56 PM">Today, 4:56 PM</div>

</div>

<div class="apple-rss-article-body-container">

<div class="apple-rss-article-body">

A southern California man killed by a grizzly bear in Alaska's backcountry was shooting photos of the animal that killed him just moments before the attack, a National Park Service official said Sunday.

… <!-- end article --></div>

Example code: See Daily Lift source files: RSSFeed.h and RSSFeed.m

### Server database

Table: MYSQL database located on my Rayoflight windows server that I rent. Database name: mysql2905int.domain.com

Table name: bingengine with columns: id, url, name, description

### Web services

The code below is just a sample. See entire app SearchEngine to see the service working. The app creates bookmarks of URL’s and when selected opens the URL into a web page. The key is the bookmarks are shared. So if two people are running the app the list of bookmarks are shared (and refresh when the app is opened).

***Read data***

<?php

$con = mysql\_connect("mysql2905int.domain.com","u1124564\_zeeshan","test123321");

if (!$con)

{

die('Could not connect: ' . mysql\_error());

}

mysql\_select\_db("db1124564\_searchengine", $con);

$result = mysql\_query("SELECT \* FROM bingengine");

echo "<records>";

while($row = mysql\_fetch\_array($result))

{

echo "<record>";

echo "<id>" . $row['id'] . "</id>" . "<name>" . $row['name'] . "</name>";

echo "</record>";

}

echo "</records>";

mysql\_close($con);

?>

**Called from iPhone code**

-(void) loadData{

NSString \*reqURL = [NSString stringWithFormat:@"http://rayoflightsoftware.com/iosservices/Tselect.php"];

NSURL \*url = [NSURL URLWithString:reqURL];

ASIHTTPRequest \*request = [ASIHTTPRequest requestWithURL:url];

[request setDelegate:self];

[request setDidFinishSelector:@selector(requestSelect:)];

[request startAsynchronous];

[uactivity startAnimating];

lblFilterName.hidden = FALSE;

}

given in .h:

NSMutableArray \*arrData;

-(void)requestSelect:(ASIHTTPRequest \*)request{

//clear data if previous

[arrData removeAllObjects];

NSData \*responseXML = [request responseData];

CXMLDocument \*doc = [[[CXMLDocument alloc] initWithData:responseXML options:0 error:nil] autorelease];

NSArray \*nodes = [doc nodesForXPath:@"//record" error:nil];

for (CXMLElement \*nodeCat in nodes) {

NSMutableDictionary \*row = [[NSMutableDictionary alloc] init];

NSLog(@"%@",[[nodeCat childAtIndex:0] stringValue]);

NSLog(@"%@",[[nodeCat childAtIndex:1] stringValue]);

NSLog(@"%@",[[nodeCat childAtIndex:2] stringValue]);

NSLog(@"%@",[[nodeCat childAtIndex:3] stringValue]);

//id

[row setValue:[[nodeCat childAtIndex:0] stringValue] forKey:@"id"];

//name

[row setValue:[[nodeCat childAtIndex:1] stringValue] forKey:@"name"];

[arrData addObject:row];

[row release];

}

[tblView reloadData];

[uactivity stopAnimating];

lblFilterName.hidden = TRUE;

}

**Delete a record**

<?php

$con = mysql\_connect("mysql2905int.domain.com","u1124564\_zeeshan","test123321");

if (!$con)

{

die('Could not connect: ' . mysql\_error());

}

mysql\_select\_db("db1124564\_searchengine", $con);

mysql\_query("DELETE FROM bingengine WHERE id='$\_POST[id]'");

mysql\_close($con);

?>

//Delete rows images and folders

- (void)tableView:(UITableView \*)tableView commitEditingStyle:(UITableViewCellEditingStyle)editingStyle forRowAtIndexPath:(NSIndexPath \*)indexPath{

NSMutableDictionary \*dicrow = [arrData objectAtIndex:indexPath.row];

if (editingStyle == UITableViewCellEditingStyleDelete) {

ASIFormDataRequest \*request = [[ASIFormDataRequest alloc]

initWithURL:[NSURL URLWithString:@"http://rayoflightsoftware.com/iosservices/Tdelete.php"]];

[request setPostValue:[dicrow objectForKey:@"id"] forKey:@"id"];

[request setDelegate:self];

[request setDidFinishSelector:@selector(requestDelete:)];

[request startAsynchronous];

[uactivity startAnimating];

lblFilterName.hidden = FALSE;

}

}

**Insert a record**

<?php

$con = mysql\_connect("mysql2905int.domain.com","u1124564\_zeeshan","test123321");

if (!$con)

{

die('Could not connect: ' . mysql\_error());

}

mysql\_select\_db("db1124564\_searchengine", $con);

mysql\_query("INSERT INTO bingengine (url, name, description)

VALUES ('$\_POST[url]', '$\_POST[name]', '$\_POST[description]')");

mysql\_close($con);

?>

ASIFormDataRequest \*request = [[ASIFormDataRequest alloc]

initWithURL:[NSURL URLWithString:@"http://rayoflightsoftware.com/iosservices/Tinsert.php"]];

[request setPostValue:txtUrl.text forKey:@"url\_detail"];

[request setPostValue:txtName.text forKey:@"name"];

[request setPostValue:txtDescription.text forKey:@"des\_detail"];

[request setDelegate:self];

[request setDidFinishSelector:@selector(requestDelete:)];

[request startAsynchronous];

Problem set #8: Load and run the code to connect to the server above

OR

within our development environment create a table view that loads data from a MYSQL database. You can use any data but make sure at least 10 rows are loaded and at least 2 pieces of information are displayed from the database table.

Bonus: create a function to insert new records and delete rows based on input from an input screen.

## 9. Using device tools

### Orientation

The standard method interfaceOrientation gets called when users rotate the device and in general you return YES as below to allow all orientations. This can be limited to just valid orientations if needed. Also other methods are available to respond to orientation changes…see documentation.

- (BOOL)shouldAutorotateToInterfaceOrientation:(UIInterfaceOrientation)interfaceOrientation {

// Return YES for supported orientations

return YES;

}

### Media

iOS can stream audio or video content. See Cookbook chapter 9: page 483

### Email

iOS can send email with text and file attachments. It will bring up an appropriate screen

In .h:

#import <MessageUI/MessageUI.h>

@interface about : UIViewController <MFMailComposeViewControllerDelegate> …

in .m:

-(void)doSendEmail {

if ([MFMailComposeViewController canSendMail]) {

NSDateFormatter \*dateFormat = [[NSDateFormatter alloc] init];

[dateFormat setDateFormat:@"MMM d yyyy"];

NSString \*adate = [dateFormat stringFromDate:[[NSDate alloc] init]];

[dateFormat release];

MFMailComposeViewController\* controller = [[MFMailComposeViewController alloc] init];

controller.mailComposeDelegate = self;

[controller setSubject:[NSString stringWithFormat:@"Registration for %@", aName.text]];

NSString \*amsg=[NSString stringWithFormat:@"Thank you for purchasing PracTrak. Please send this email to register. Visit www.rayoflightsoftware.com for documentation and support. Registration on %@ for %@ %@ using email: %@",adate,aName.text,aDegree.text,self.email\_addr];

[controller setMessageBody:amsg isHTML:NO];

[controller setToRecipients:[NSArray arrayWithObjects:@"registration@rayoflightsoftware.com", nil]];

is\_dobackup=0;

[self presentModalViewController:controller animated:YES];

[controller release];

}

// page=0;

// afterEmail=1;

}

- (void)mailComposeController:(MFMailComposeViewController\*)controller

didFinishWithResult:(MFMailComposeResult)result

error:(NSError\*)error;

{

if (result == MFMailComposeResultSent) {

if (is\_dobackup==0) {

aName.hidden=YES;

aRate.hidden=YES;

aDegree.hidden=YES;

lName.hidden=YES;

lRate.hidden=YES;

aemail.hidden=YES;

lemail.hidden=YES;

pwd\_on.hidden=YES;

page=0;

afterEmail=1;

}

else {

utils \*autil = [[utils alloc] init];

[autil setLatestBackup:0 :1];

[autil release];

is\_dobackup=0;

}

}

[self dismissModalViewControllerAnimated:YES];

}

### Date formatting

NSDateFormatter \*dateFormat = [[NSDateFormatter alloc] init];

[dateFormat setDateFormat:@"MM"];

NSDate \*today = [[NSDate alloc] init];

int currMonth = [[dateFormat stringFromDate:today] intValue];

[dateFormat release];

[today release];

### Maps and Location

Cookbook chapter 6: page 401

### Gestures

Cookbook chapter 7: page 427

### Address book

Cookbook chapter 10: page 513

### Camera and Photo Library

Cookbook chapter 11: page 547

### Graphics and Animation

Cookbook chapter 15: page 693

Problem set #9: a) Create a new app that plays a local MP3 audio file. Add an email function to the app you create that will email a note to you every time the audio file finishes playing. Add one more function to the app that uses either a map, media, camera, address book, or graphics function described above.

### Provisioning

This course will not require you to deploy apps on other devices until the final project as the iOS iPhone/iPad simulator can be used for most development testing. The description below is given to describe the provisioning and deployment process as a way of background of how these occur when you are ready to deploy apps on real devices for testing or distribution purposes.

Once debugged in simulator you should deploy on local iPhone/iPad by creating a provision file. Provisioning is set up via the Apple Developer site and the instructor will need to create a provision file for each app to be deployed. It is suggested that most work occur within the simulator until the final testing of the app is required.

Note the provisioning file created must be double-clicked on the development machine to register it with the development platform and within the app it must be configured to use the provision file within the Project, Build Settings, Code Signing area.

Info.plist: this file contains critical flags and settings that must be set for deployment. See Apple documentation for list of settings needed. For example, if the app will be accessing the web or streaming video or audio special settings are needed. Also check the project screen for settings up all icons, splash screens and other graphics.

### Distributing on device(s)

In debug mode with a valid provisioning file XCode development environment can deploy an app to a registered device directly. To distribute the app to other users without a development environment the following actions can be taken to distribute through iTunes:

An ipa app file must be created by a) setting the device to iOS Device and then b) under the Product menu select Clean then Build.

The ipa app file and provision file are deployed to a device that is attached automatically through development platform or via drag/drop in iTunes. If others are testing you can email them the two files to drag/drop in iTunes as long as their devices’ UDID is in your provisioning file.