iPhone App Dev

Lesson 9

Source Codes

https://github.com/makzan/ios-dev-course-example

Contact

makzan@42games.net

Summary

- Usage of Tab Bar
- Creating Our Tabbed Application
- Ordering and Deleting Table View Cell
- Playing Sound Effects

Usage of Tab Bar

Usage of Tab Bar

Quick switch between utilities features

Usage of Tab Bar

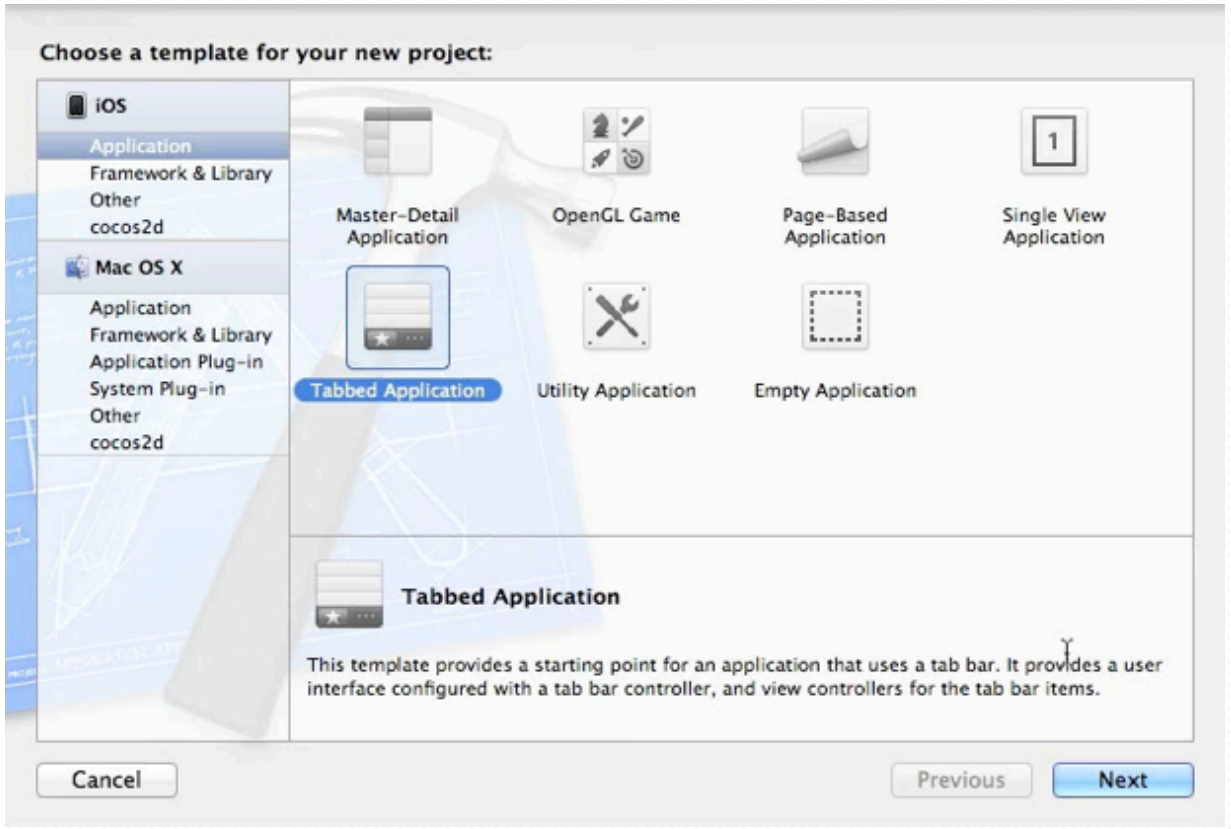
Main navigation between views

Usage of Tab Bar

Non-standard tab bar

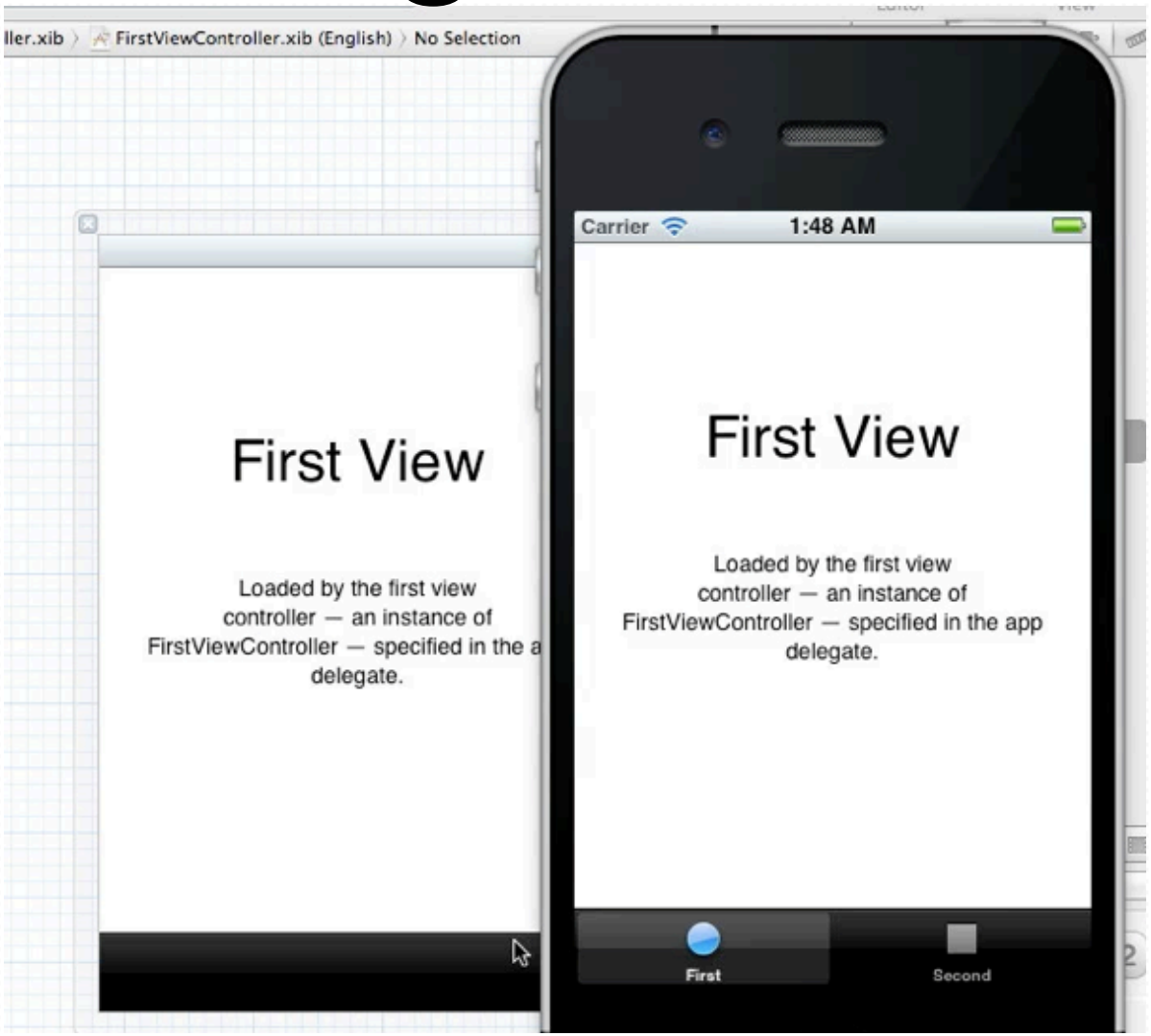
Create our Tabbed Application

Creating a Tabbed App



Create a new project with "Tabbed Application" preset.

Creating a Tabbed App



The preset setup everything and just run it to try the two views tabbed app.

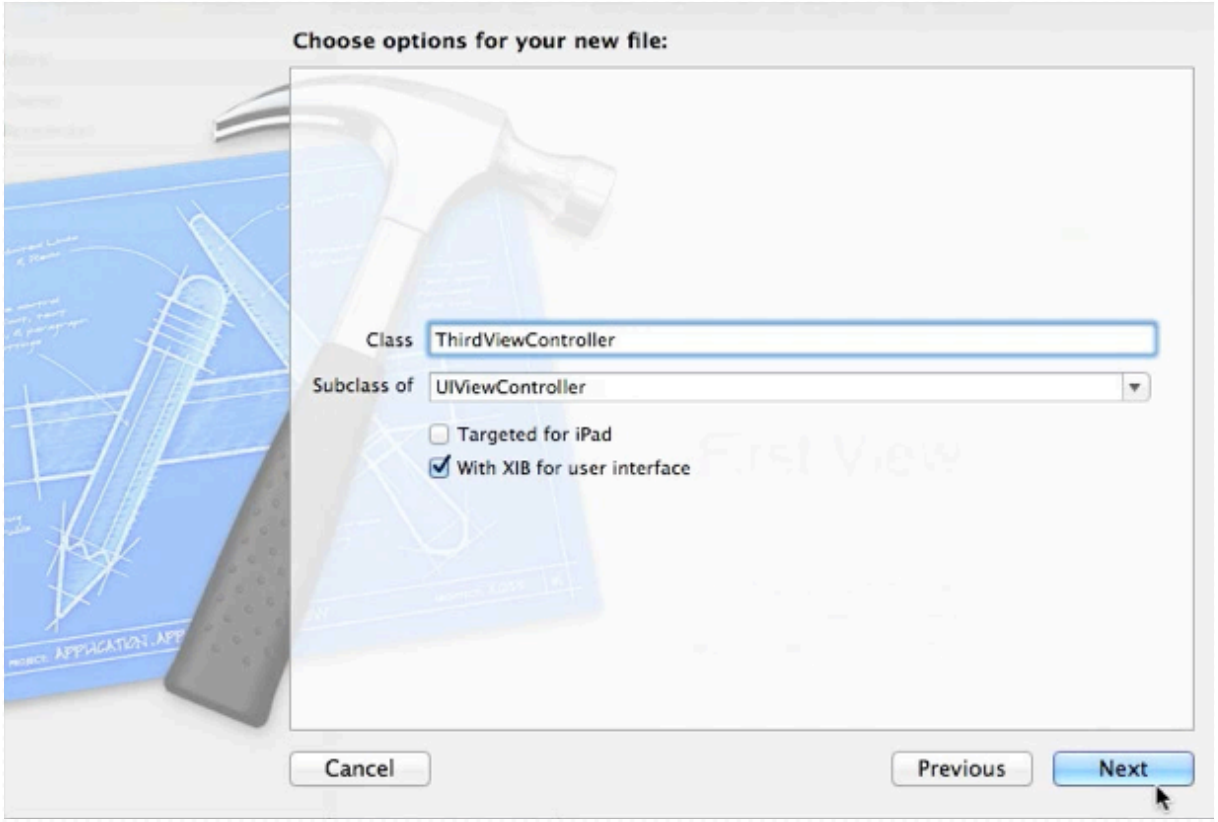
Creating a Tabbed App

AppDelegate.m

```
- (BOOL)application:(UIApplication *)application
    didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    self.window = [[[UIWindow alloc] initWithFrame:[[UIScreen mainScreen]
        bounds]] autorelease];
    // Override point for customization after application launch.
    UIViewController *viewController1 = [[[FirstViewController alloc]
        initWithNibName:@"FirstViewController" bundle:nil] autorelease];
    UIViewController *viewController2 = [[[SecondViewController alloc]
        initWithNibName:@"SecondViewController" bundle:nil] autorelease];
    self.tabBarController = [[[UITabBarController alloc] init] autorelease];
    self.tabBarController.viewControllers = [NSArray arrayWithObjects:
        viewController1, viewController2, nil];
    self.window.rootViewController = self.tabBarController;
    [self.window makeKeyAndVisible];
    return YES;
}
```

XCode generates the above codes for us.

Creating a Tabbed App



Let's add the third view to the app

Creating a Tabbed App

AppDelegate.m

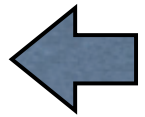
```
//  AppDelegate.m
//  TabDemo
//
//  Created by Freshman on 6/30/12.
//  Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//

#import "AppDelegate.h"

#import "FirstViewController.h"

#import "SecondViewController.h"

#import "ThirdViewController.h

@imple
ThirdViewController.h
ThirdViewController.m
ThirdViewController.x…

@synth
@synthesize tabBarController = _tabBarController;
```

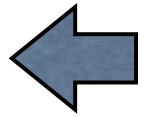
Import our new class into AppDelegate.m

Creating a Tabbed App

AppDelegate.m

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)
    launchOptions
{
    self.window = [[[UIWindow alloc] initWithFrame:[[UIScreen mainScreen] bounds]] autorelease];
    // Override point for customization after application launch.
    UIViewController *viewController1 = [[[FirstViewController alloc] initWithNibName:
        @"FirstViewController" bundle:nil] autorelease];
    UIViewController *viewController2 = [[[SecondViewController alloc] initWithNibName:
        @"SecondViewController" bundle:nil] autorelease];

    ThirdViewController *viewController3 = [[[ThirdViewController alloc] initWithNibName:
        @"ThirdViewController" bundle:nil] autorelease];

    self.tabBarController = [[[UITabBarController alloc] init] autorelease];

    self.tabBarController.viewControllers = [NSArray arrayWithObjects:viewController1,
        viewController2, viewController3, nil];

    self.window.rootViewController = self.tabBarController;
    [self.window makeKeyAndVisible];
    return YES;
}
```

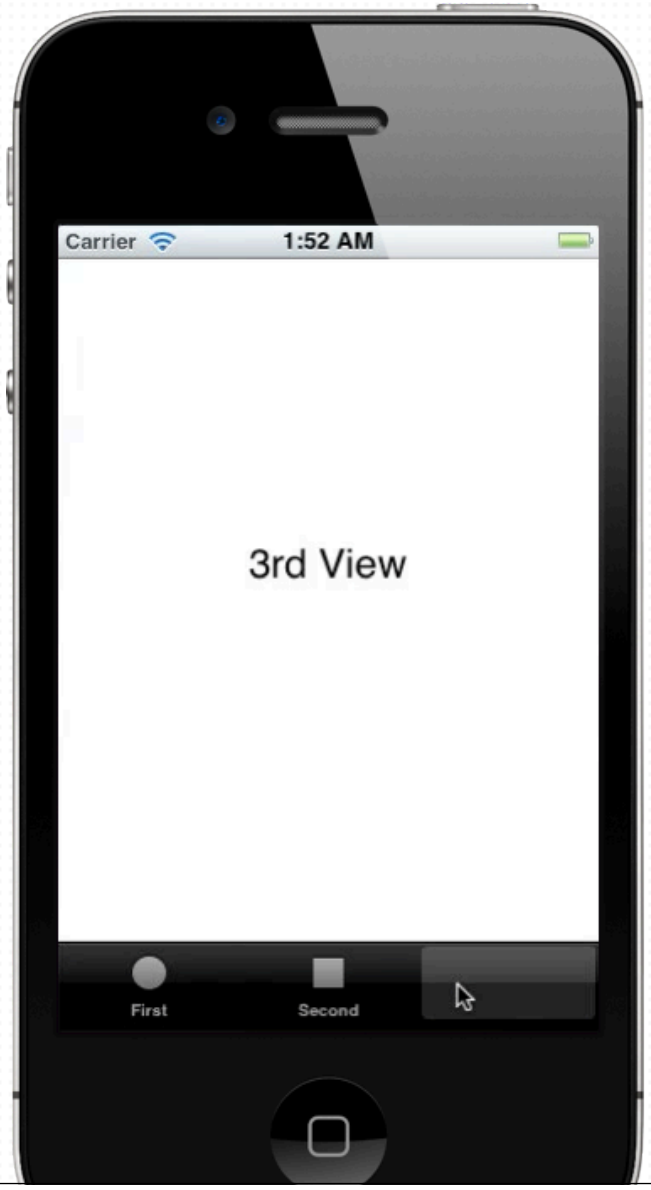




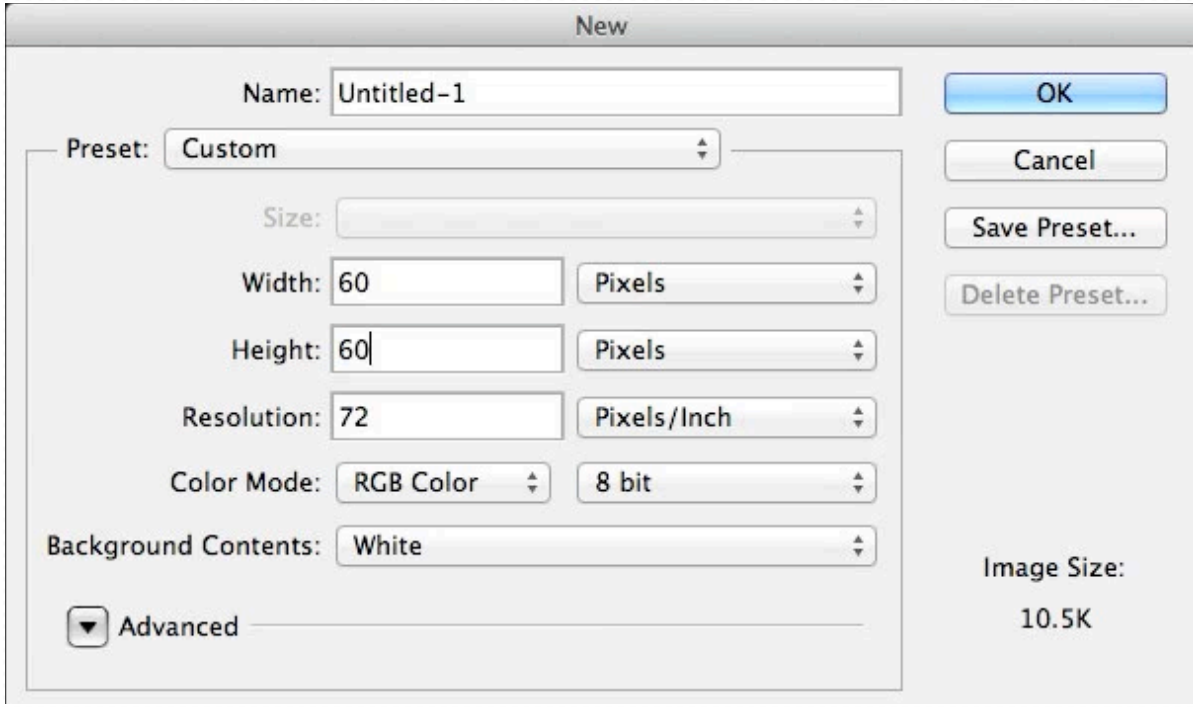
Add the ThirdViewController instance to TabBarController

Creating a Tabbed App

Let's run it and we have another tab without neither icons nor title.

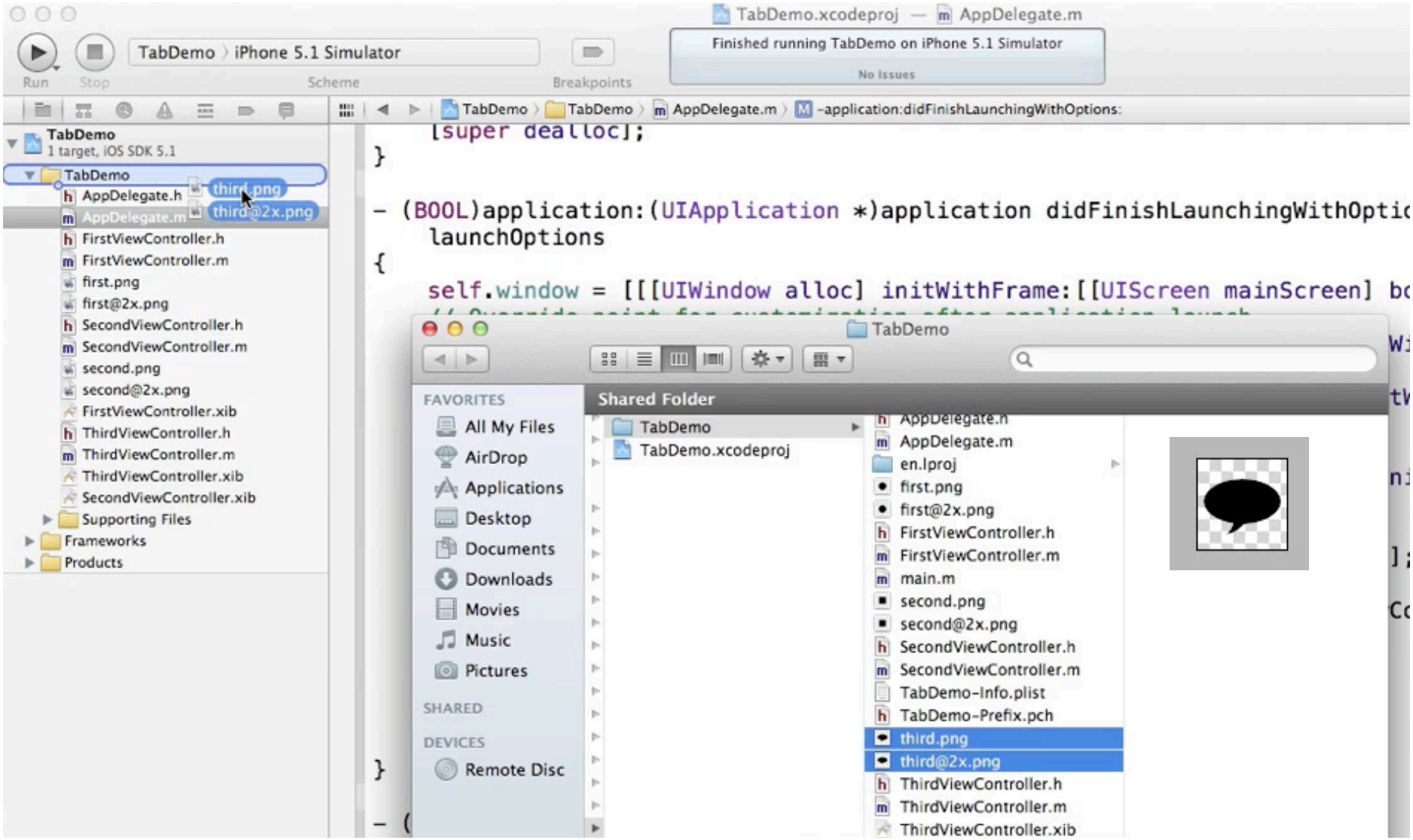


Tab Bar Icon



The tab bar icon is 30x30 points.
We need a 60x60 points image for retina display.
Create the icon and export it to
third.png and **third@2x.png**

Tab Bar Icon



Import the icon files into the project

Tab Bar Icon

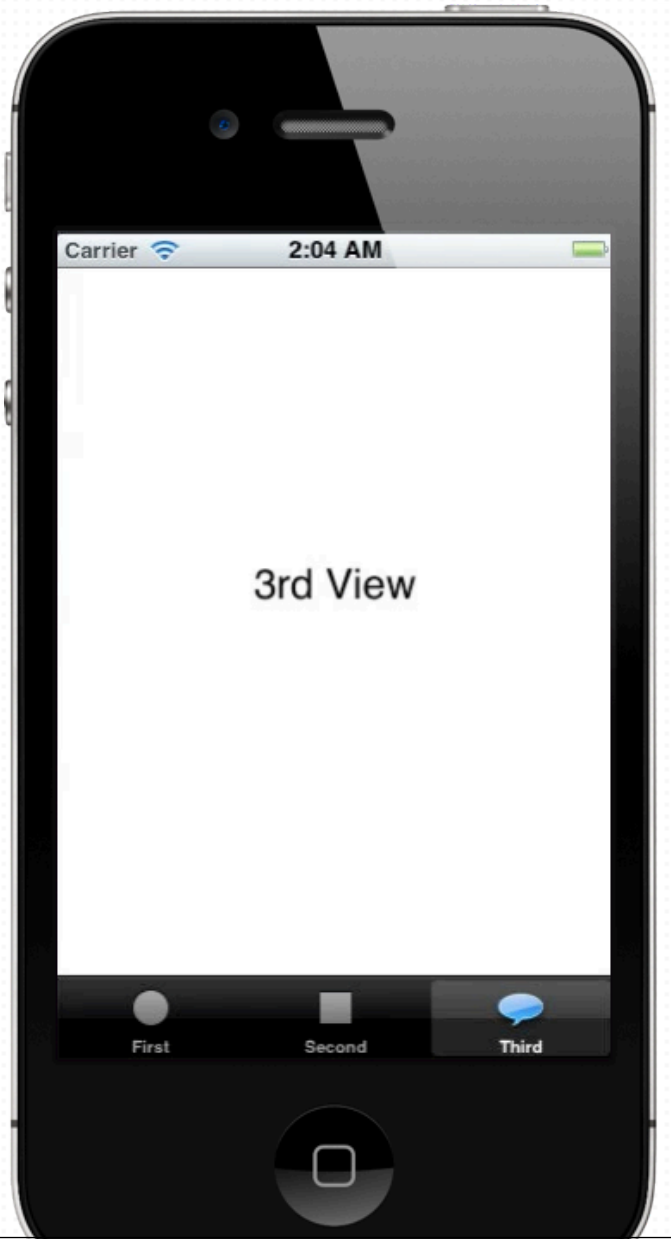
ThirdViewController.m

```
- (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil
{
    self = [super initWithNibName:nibNameOrNil bundle:nibBundleOrNil];
    if (self) {
        // Custom initialization

        self.title = @"Third";
        self.tabBarItem.image = [UIImage imageNamed:@"third"];
    }
    return self;
}
```

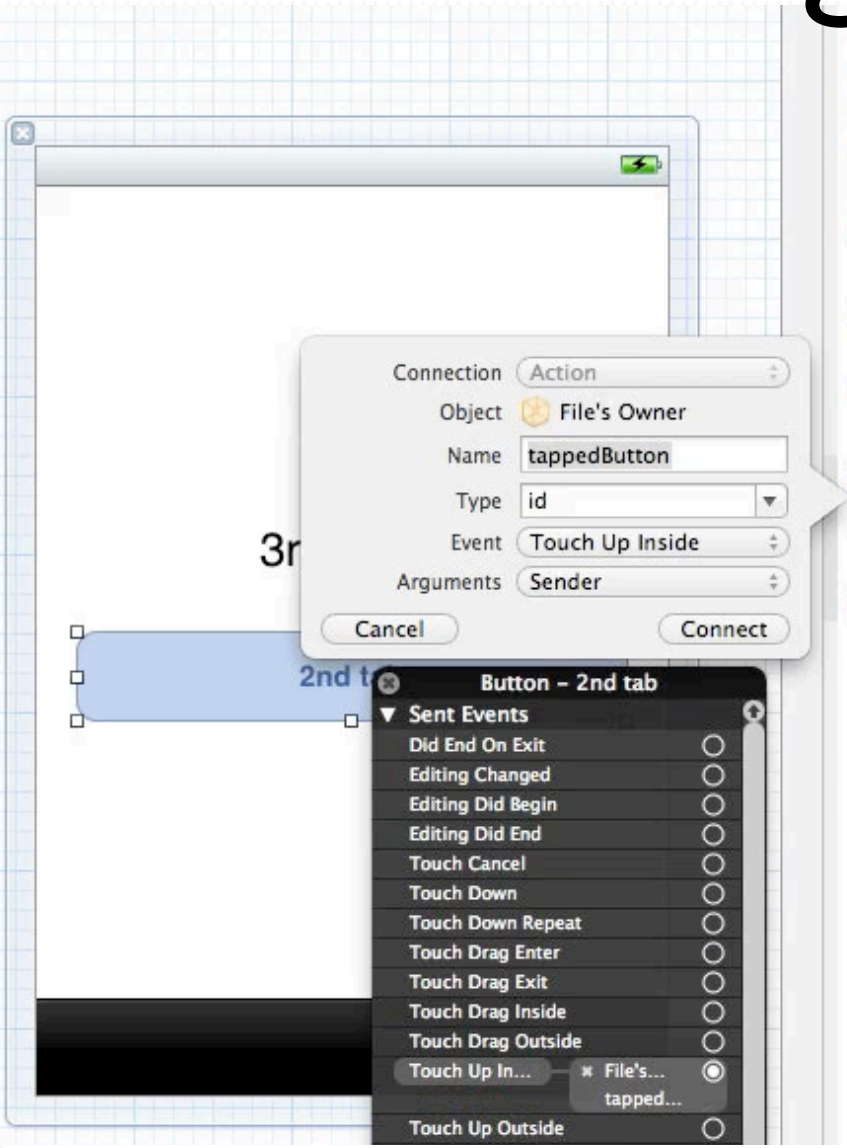
When we init the tabbed view controller, we set the title and image of tabBarItem for the tab.

Tab Bar Icon

Our new tab with title and icon



Switching Tab in Code




```
//
//  ThirdViewController.h
//  TabDemo
//
//  Created by Freshman on 6/30/12.
//  Copyright (c) 2012 __MyCompanyName__. All rights
    reserved.
//

#import <UIKit/UIKit.h>

@interface ThirdViewController : UIViewController
- (IBAction)tappedButton:(id)sender;

@end
```

Create a button in the third view with IBAction connected.

Switching Tab in Code

ThirdViewController.m

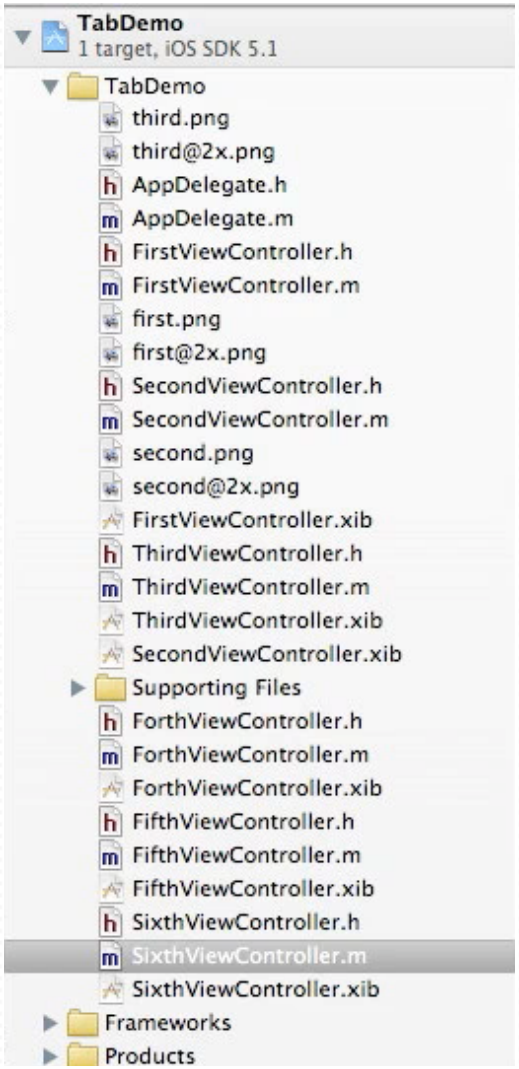
```
- (IBAction)tappedButton:(id)sender {
    self.tabBarController.selectedIndex = 1;
}
```

When the button is tapped, set the **selectedIndex** of the tab bar controller.

Switching Tab in Code

It jumps to 2nd tab
when tapped the button.

More than 5 tabs



Let's have total 6 view controllers for our tab bar.

More than 5 tabs

AppDelegate.m

```
self.tabBarController.viewControllers = [NSArray arrayWithObjects:
    viewController1, viewController2, viewController3, viewController4,
    viewController5, viewController6, nil];
```

Create the instance of those view controllers and add them into the tab bar controller.

More than 5 tabs

When there are more than 5 tabs in the tab bar controllers, a More button is automatically added.

More than 5 tabs

The extra tabs are placed into a table view inside a navigation controller.

More than 5 tabs

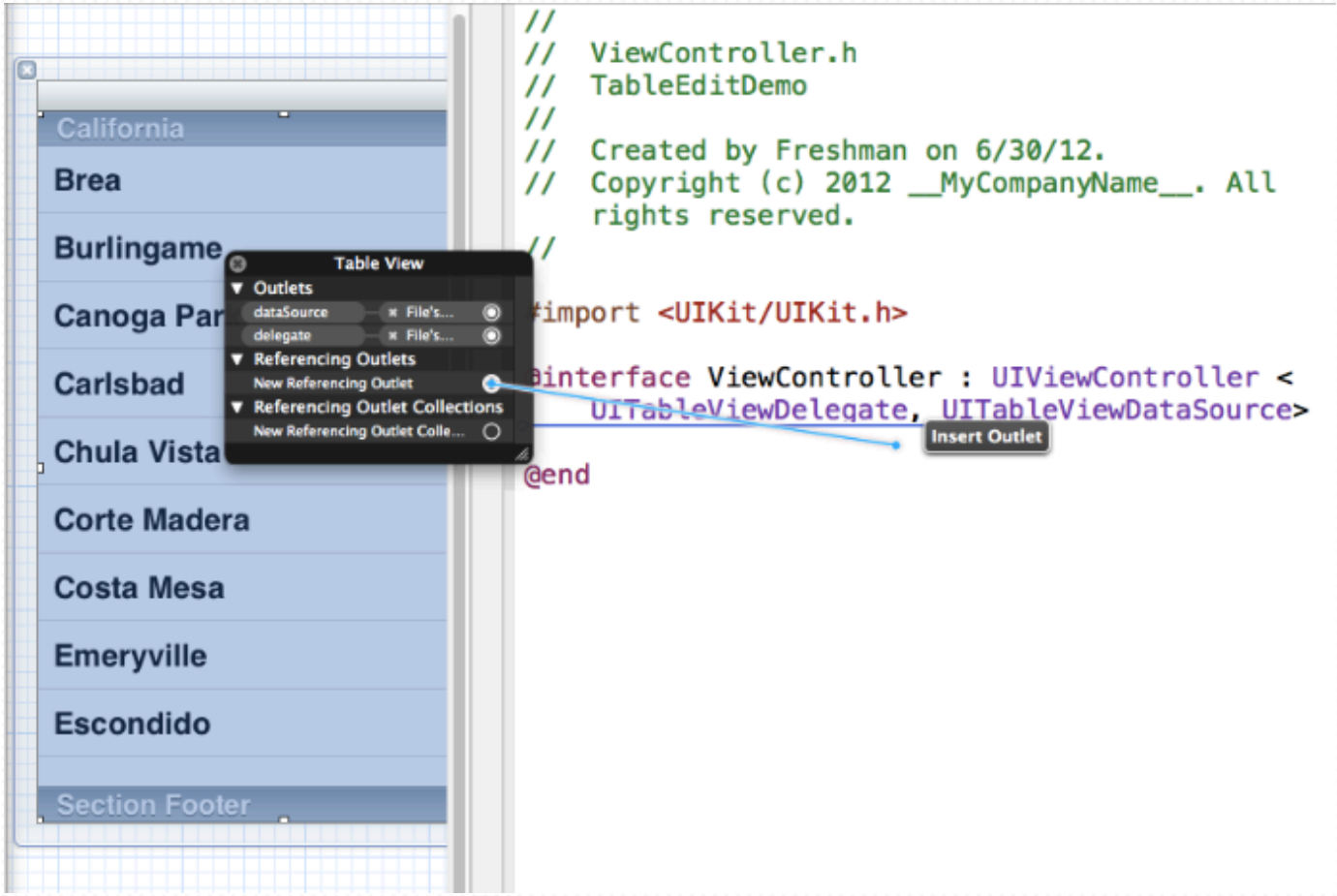
A tab arrangement view appears when we tap the automatically generated edit button in More tab.

User can decide which 4 to show at the bottom and the order of them.

Ordering and Deleting Table View Cells

Let's talk more on table view

Setup a Standard Table View



Create a new application with table view setup

Setup a Standard Table View

ViewController.m

```
- (void)viewDidLoad
{
    [super viewDidLoad];
    // Do any additional setup after loading the view, typically from a nib.

    // our data model
    self.dataArray = [[NSMutableArray alloc] initWithObjects:@"Data 1", @"Data 2", @"Data 3", @"Data 4", @"Data 5", @"Data 6"
        , @"Data 7", nil];
```

Setup the our data to be displayed on table.

Setup a Standard Table View

ViewController.m

```
- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section
{
    return [self.dataArray count];
}
```

And the number of rows from our data array.

Setup a Standard Table View

ViewController.m

```
- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath
{
    UITableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:@"Identifier"];
    if (cell == nil)
    {
        cell = [[[UITableViewCell alloc] initWithStyle:UITableViewCellStyleDefault reuseIdentifier:@"Identifier"] autorelease
            ];
    }

    cell.textLabel.text = [self.dataArray objectAtIndex:indexPath.row];

    return cell;
}
```

And a standard implementation of the table view cell delegate method.

Deleting Table View Cell

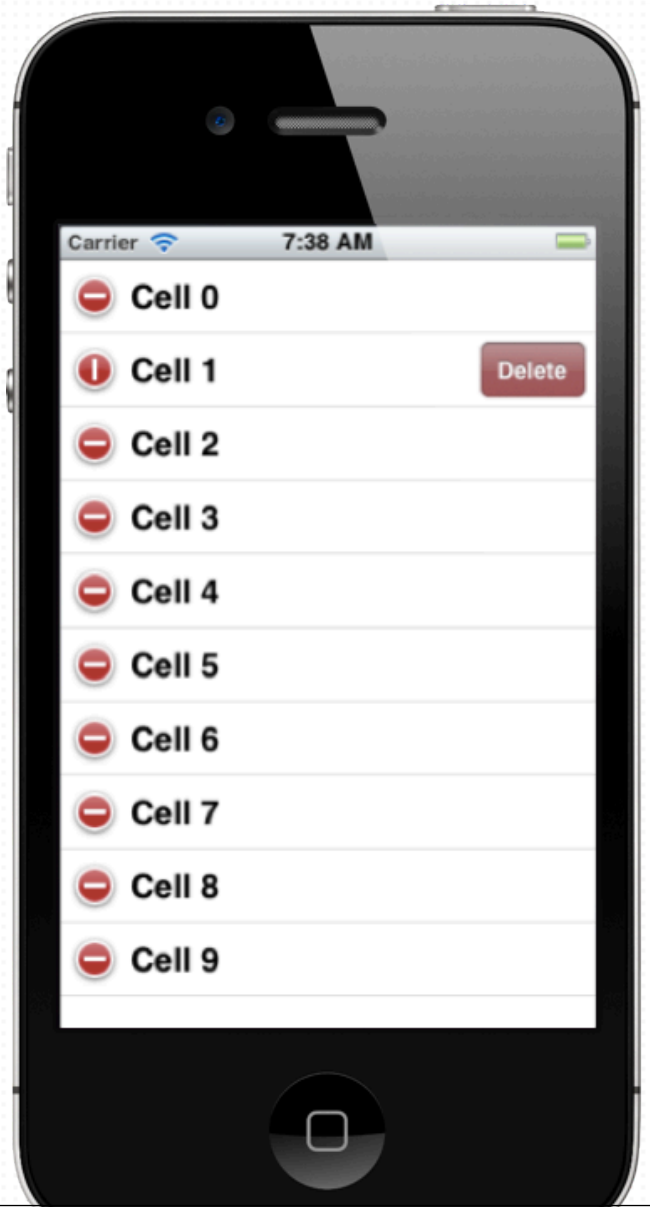
ViewController.m

```
- (void)viewDidLoad
{
    [super viewDidLoad];
    // Do any additional setup after loading the view, typically from a nib.

    [self.tableView setEditing:YES animated:YES];
}
```

Set the table view to enter Editing Mode.

Deleting Table View Cell

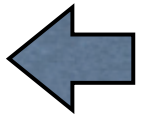
The table is in editing mode when app launches. But we can not resume to the normal view yet.

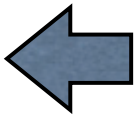


Put the Table Inside Navigation Controller

AppDelegate.m

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)
    launchOptions
{
    self.window = [[[UIWindow alloc] initWithFrame:[[UIScreen mainScreen] bounds]] autorelease];
    // Override point for customization after application launch.
    self.viewController = [[[ViewController alloc] initWithNibName:@"ViewController" bundle:nil]
        autorelease];

    UINavigationController *navController = [[[UINavigationController alloc]
        initWithRootViewController:self.viewController] autorelease];

    self.window.rootViewController = navController;
    [self.window makeKeyAndVisible];
    return YES;
}
```





Modify the AppDelegate.m to put the view controller into a navigation controller.

Toggling Editing Mode

ViewController.m

```
UIBarButtonItem *editButton = [[[UIBarButtonItem alloc] initWithBarBut
    UIBarButtonSystemItemEdit target:self action:@selector(tappedEdit)]

self.navigationItem.rightBarButtonItem = editButton;
```

In the ViewDidLoad delegate method, we create the top right button to be edit button.

When user tapped edit, we can simply toggle the editing mode of the table view.

Toggling Editing Mode

ViewController.m

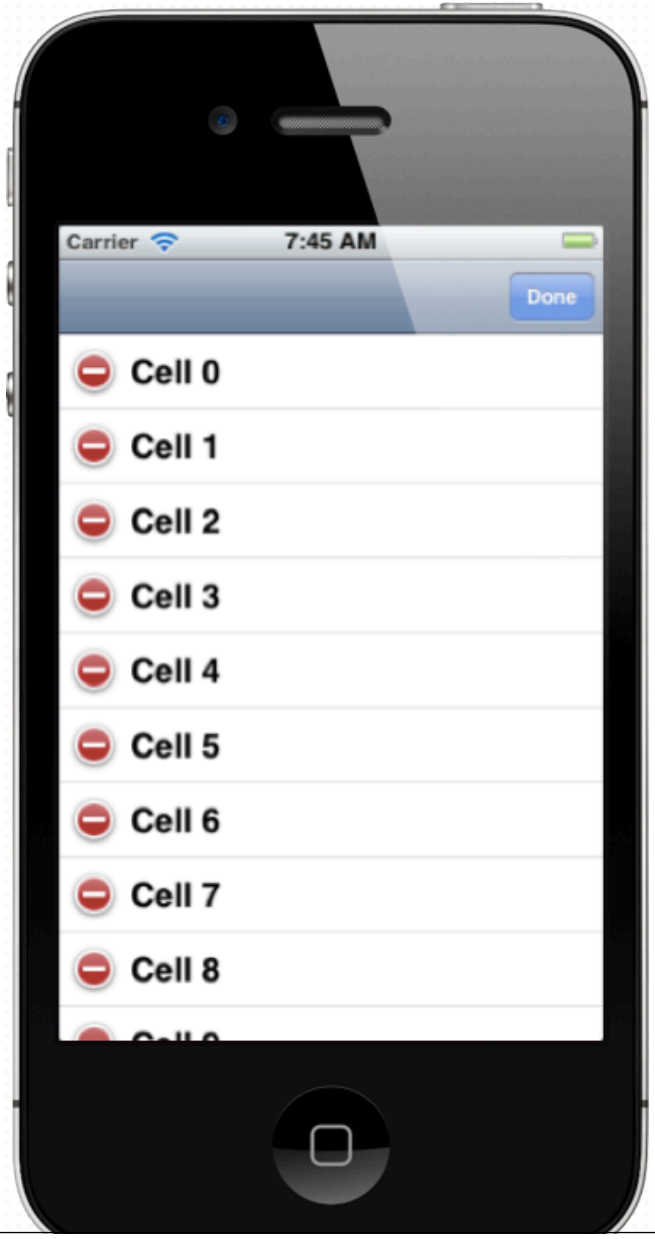
```
- (void)tappedEdit
{
    if (self.tableView.isEditing)
    {
        [self.tableView setEditing:NO animated:YES];

        // update the top right button
        UIBarButtonItem *editButton = [[[UIBarButtonItem alloc] initWithBarButtonSystemItem:UIBarButtonSystemItemEdit
                                                                                     target:self
                                                                                     action:@selector(tappedEdit)]
                                                                                    autorelease];

        self.navigationItem.rightBarButtonItem = editButton;
    }
    else
    {
        [self.tableView setEditing:YES animated:YES];


        // update the top right button
        UIBarButtonItem *doneButton = [[[UIBarButtonItem alloc] initWithBarButtonSystemItem:UIBarButtonSystemItemDone
                                                                                     target:self
                                                                                     action:@selector(tappedEdit)]
                                                                                    autorelease];


        self.navigationItem.rightBarButtonItem = doneButton;
    }

}
```

Let's toggle between editing mode and the top right button when tapping edit.

Toggling Editing Mode

Let's run our app and we can toggle between editing mode and normal view.

But the deleting feature does not work yet.



Toggling Editing Mode

ViewController.m

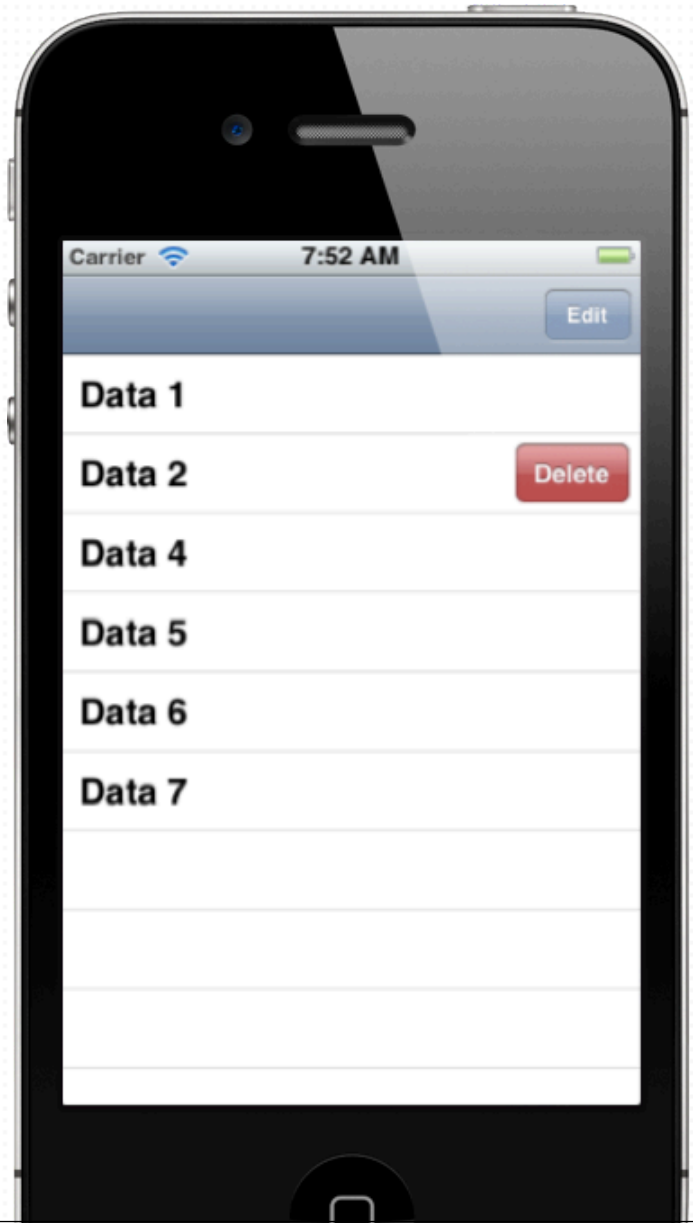
```
- (void)tableView:(UITableView *)tableView
commitEditingStyle:(UITableViewCellEditingStyle)editingStyle
forRowAtIndexPath:(NSIndexPath *)indexPath
{
    [self.dataArray removeObjectAtIndex:indexPath.row];

    [tableView deleteRowsAtIndexPaths:[NSArray arrayWithObject:indexPath]
                     withRowAnimation:UITableViewRowAnimationRight];
}
```

There is a delegate method to allow us to really commit the deleting action.

The above code remove the entry in the data array and tell the table view to remove the cell.

Toggling Editing Mode

When we allow cell deletion in our app, the table view automatically supports swipe to delete gesture.



Toggling Editing Mode

ViewController.m

```
// disable the swipe to delete function, if you need to get rid of it.
- (UITableViewCellEditingStyle)tableView:(UITableView *)tableView
            editingStyleForRowAtIndexPath:(NSIndexPath *)indexPath
{
    if (tableView.isEditing)
    {
        return UITableViewCellEditingStyleDelete;
    }

    return UITableViewCellEditingStyleNone;
}
```

Not every app needs the swipe to delete feature, we can disable it with the above delegate method.

Reordering the cells

ViewController.m

```
// allow arranging the cell order
- (void)tableView:(UITableView *)tableView
moveRowAtIndexPath:(NSIndexPath *)sourceIndexPath
       toIndexPath:(NSIndexPath *)destinationIndexPath
{

}
```

The ordering feature is not enabled by default. It is enabled once we implemented the **moveRowAtIndexPath:toIndexPath:** delegate.

Reordering the cells

We can now re-order the table cell. But we have not handled the data logic for it.

Reordering the cells

ViewController.m

```
// allow arranging the cell order
- (void)tableView:(UITableView *)tableView
moveRowAtIndexPath:(NSIndexPath *)sourceIndexPath
      toIndexPath:(NSIndexPath *)destinationIndexPath
{
    // cache the cell content before removing it from the data array.
    id cellContent = [self.dataArray objectAtIndex:sourceIndexPath.row];

    // remove the cell content in the data array.
    [self.dataArray removeObjectAtIndex:sourceIndexPath.row];

    // re-insert the cell content into the new position in the array.å
    [self.dataArray insertObject:cellContent atIndex:destinationIndexPath.row];

    NSLog(@"%@", self.dataArray);
}
```

Let's add the data logic when we move the rows. We delete the old entry and insert it into another place in the data array.

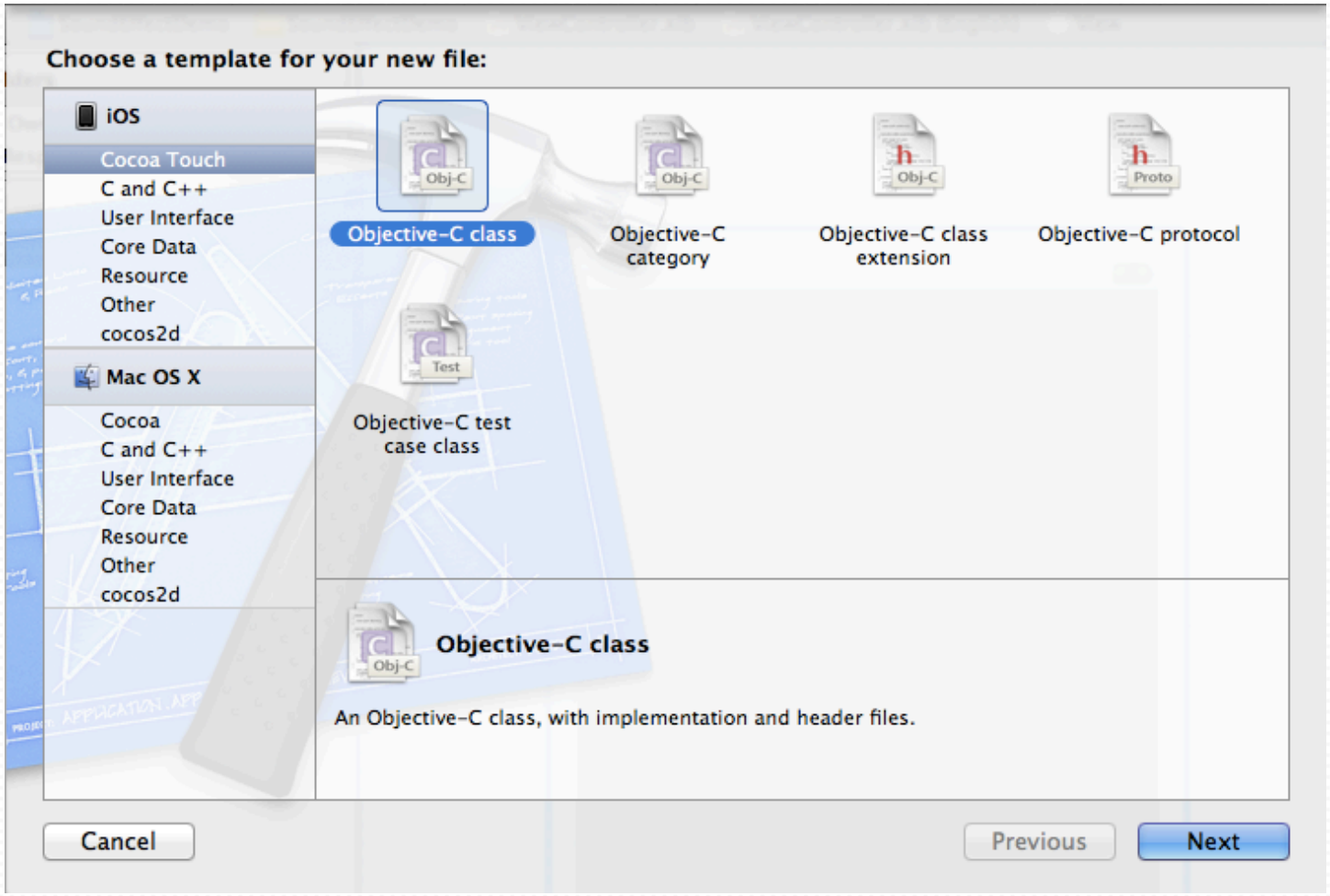
Reordering the cells

We print the array data out so we know the logic works.

Playing Sound Effects

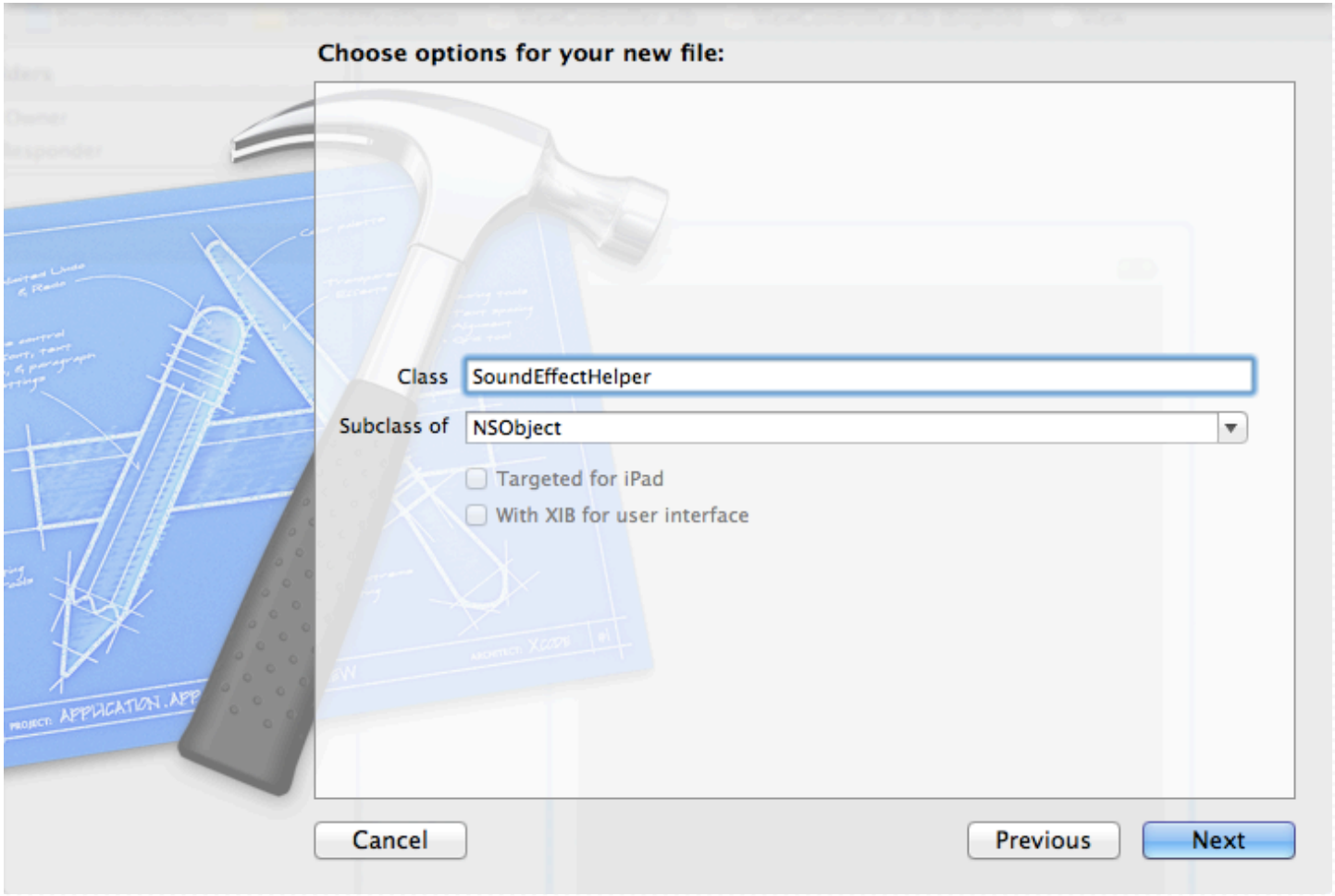
We will explore two approaches to play sounds

Playing Sound Effects



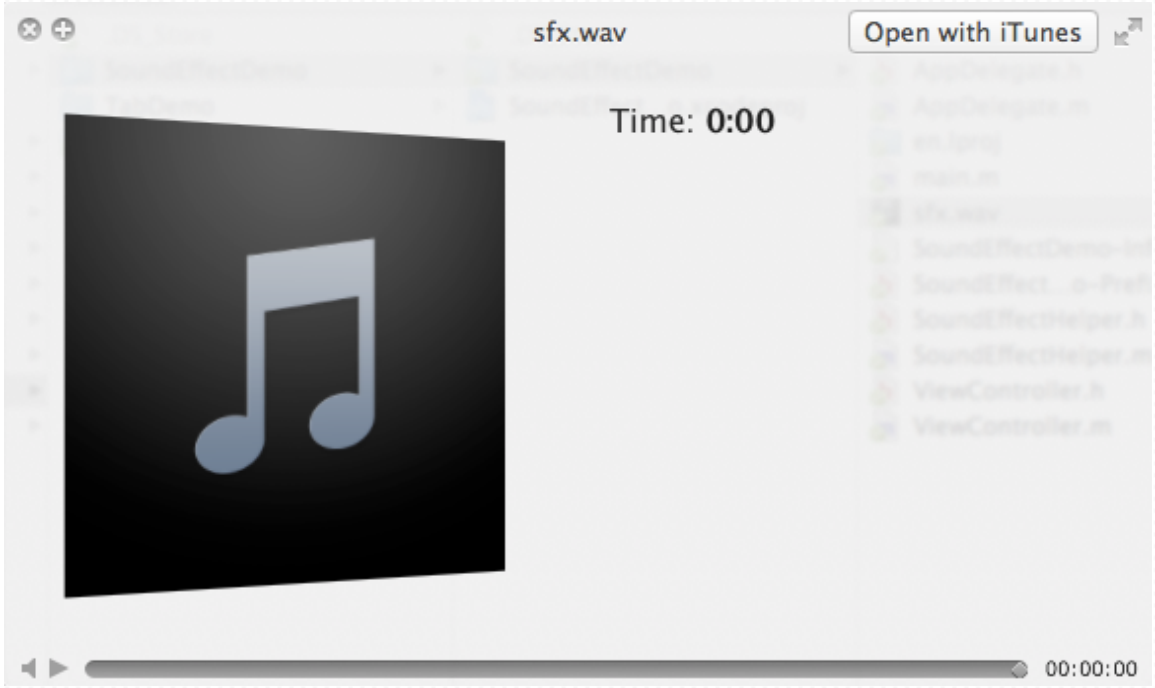
Let's create a new class dedicated for the sound effects.

Playing Sound Effects



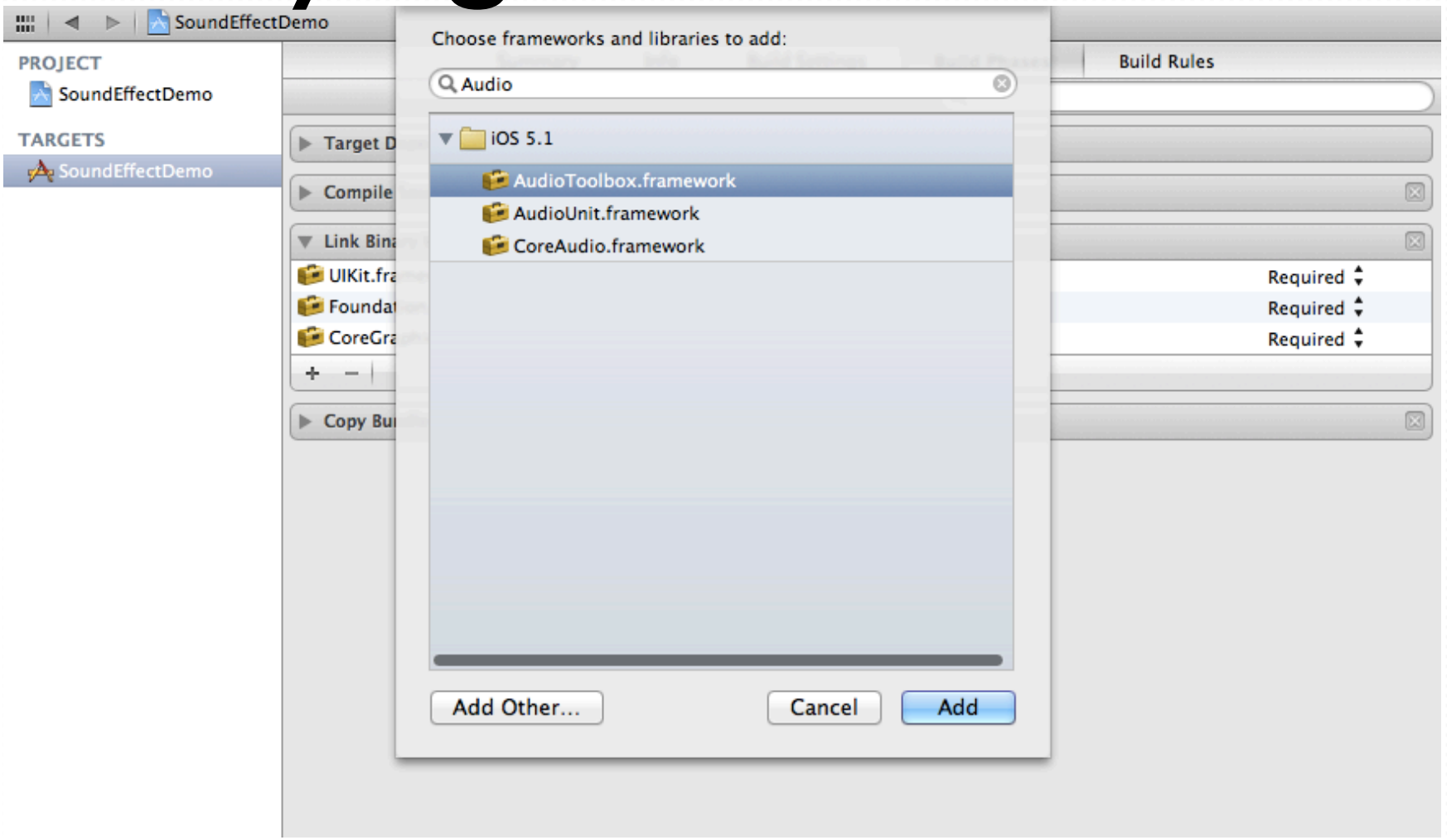
Give the class a name and set it to be subclass of NSObject.

Playing Sound Effects



We need a sound file for our demo.
Here we have a sfx.wav

Playing Sound Effects



The first approach requires the AudioToolbox framework

Playing Sound Effects

SoundEffectHelper.m

```
#import "SoundEffectHelper.h"

#import <AudioToolbox/AudioToolbox.h>

@implementation SoundEffectHelper

+ (void)playSFX
{
    SystemSoundID soundID;

    NSString* soundPath = [[NSBundle mainBundle] pathForResource:@"sfx" ofType:@"wav"];
    CFURLRef soundFileRef = (CFURLRef)[[NSURL alloc] initFileURLWithPath:soundPath];

    AudioServicesCreateSystemSoundID(soundFileRef, &soundID);

    AudioServicesPlaySystemSound(soundID);
}

@end
```

AudioToolbox provides the system audio service to play any short duration sounds.

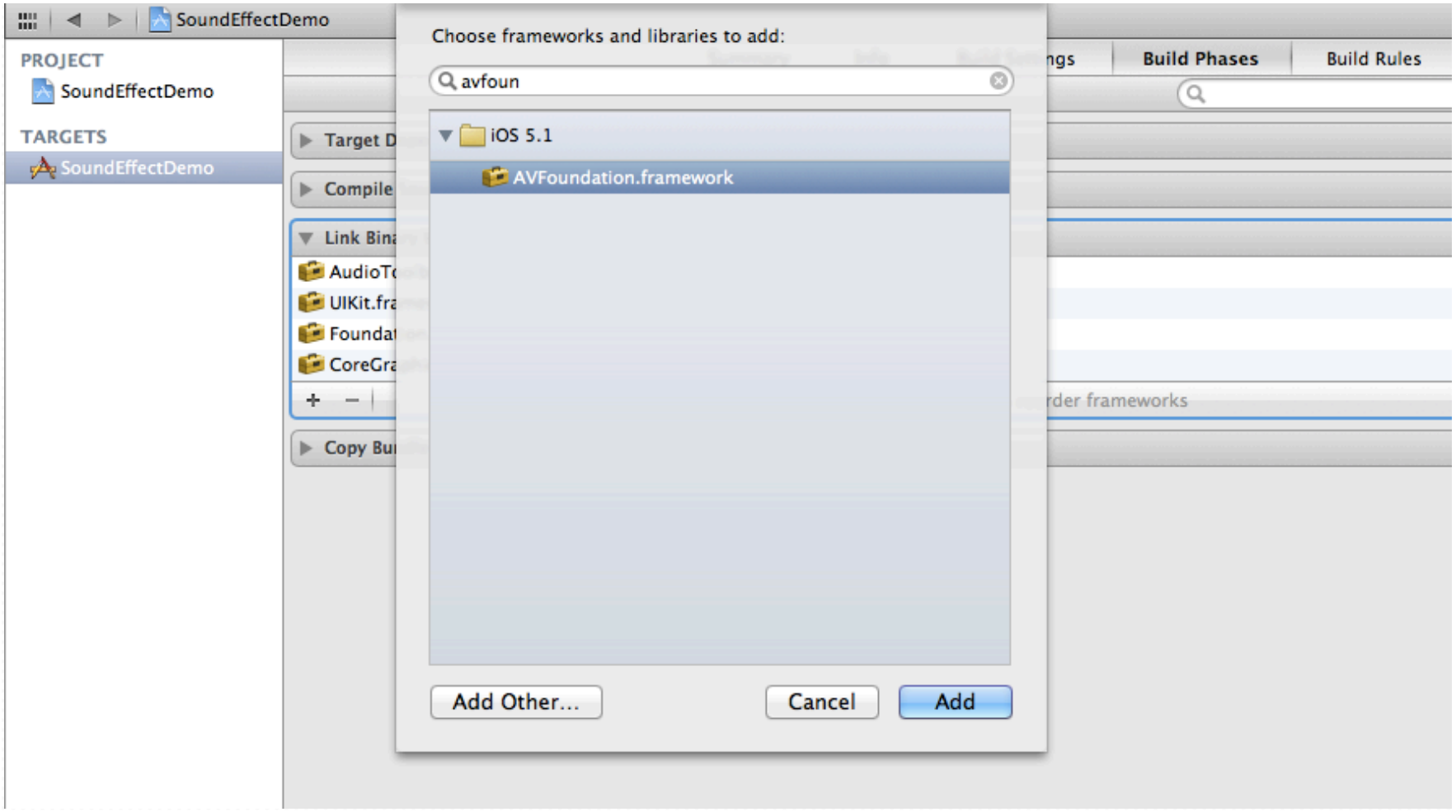
Playing Sound Effects

SoundEffectHelper.h

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

@interface SoundEffectHelper : NSObject

+ (void)playSFX;

@end
```

Don’t forget to put the method declaration on the header file for other classes to use.

Playing Sound Effects



And the 2nd approach requires the AVFoundation framework.

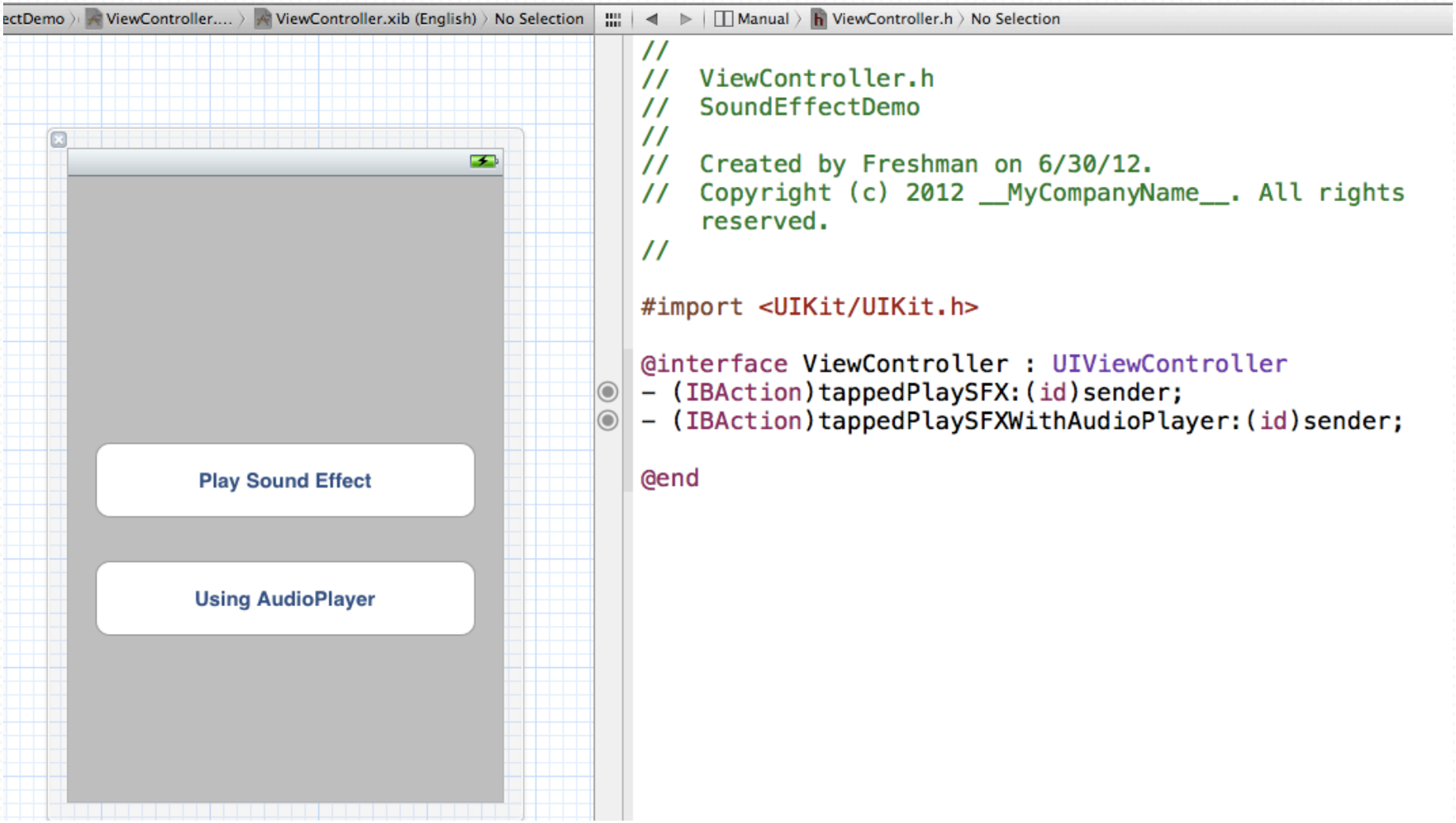
Playing Sound Effects

SoundEffectHelper.m

```
+ (void)playSFXByAudioPlayer
{
    NSString* soundPath = [[NSBundle mainBundle] pathForResource:@"sfx" ofType:@"wav"];

    NSURL *url = [NSURL fileURLWithPath:soundPath];

    AVAudioPlayer *player = [[AVAudioPlayer alloc] initWithContentsOfURL:url error:nil];

    [player play];
}
```

This time we use the AVAudioPlayer to load the audio file and play it ourselves.

Playing Sound Effects



Let's try our two methods with a testing view and two IBAction buttons

Playing Sound Effects

ViewController.m

```
- (IBAction)tappedPlaySFX:(id)sender {
    [SoundEffectHelper playSFX];
}

- (IBAction)tappedPlaySFXWithAudioPlayer:(id)sender {
    [SoundEffectHelper playSFXByAudioPlayer];
}
```

When tapped the button, we use either approach to play the sound effect.

```objc
#import <AudioToolbox/AudioToolbox.h>

#import <AVFoundation/AVFoundation.h>

@implementation SoundEffectHelper

+ (void)playSFX
{
    SystemSoundID soundID;

    // get the filepath from filename.
    NSString* soundPath = [[NSBundle mainBundle] pathForResource:@"sfx" ofType:@"wav"];

    // create CFURLRef from the path, which we will use for AudioServices
    CFURLRef soundFileRef = (CFURLRef)[[NSURL alloc] initFileURLWithPath:soundPath];

    AudioServicesCreateSystemSoundID(soundFileRef, &soundID);

    AudioServicesPlaySystemSound(soundID);
}

+ (void)playSFXByAudioPlayer
{
    NSString* soundPath = [[NSBundle mainBundle] pathForResource:@"sfx" ofType:@"wav"];

    NSURL *url = [NSURL fileURLWithPath:soundPath];

    AVAudioPlayer *player = [[AVAudioPlayer alloc] initWithContentsOfURL:url error:nil];

    [player play];
}

@end
```

SoundEffectHelper.m

Vibrate the phone

```
+ (void)vibrate
{
    AudioServicesPlaySystemSound(kSystemSoundID_Vibrate);
}
```

If the iOS device supports vibration, we can use the AudioToolbox to make the vibrate with the above code.

Playing Sound Effects

- AudioToolbox is good for playing short duration sounds. It doesn't provide much control but it is straightforward to use.
- AVFoundation let us control how we play the sounds. It is powerful but may be difficult to control.

Playing Sound Effects

```
+ (void)playSFXByAudioPlayer
{
    NSString* soundPath = [[NSBundle mainBundle] pathForResource:@"sfx" ofType:@"wav"];

    NSURL *url = [NSURL fileURLWithPath:soundPath];

    AVAudioPlayer *player = [[AVAudioPlayer alloc] initWithContentsOfURL:url error:nil];

    // prevent the AudioPlayer interruping the iPod music playing.
    [[AVAudioSession sharedInstance] setCategory:AVAudioSessionCategoryAmbient error:nil];

    [player play];
}
```

One more thing, we can control how the sound interrupts the other system sounds with AVAudioSession

Next Lesson

- We will have a quiz.
- We will talk about deploying the app.
- Everyone will get 2-3 minutes to talk about your ideas