iPhone App Dev

Lesson 9

Source Codes

https://github.com/makzan/ios-dev-course-example-2013

Contact

makzan@42games.net

Summary

- Usage of Tab Bar
- Creating Our Tabbed Application
- Creating Custom Cell in Table View
- Playing Sound Effects

Usage of Tab Bar

Usage of Tab Bar

Quick switch between utilities features

Usage of Tab Bar

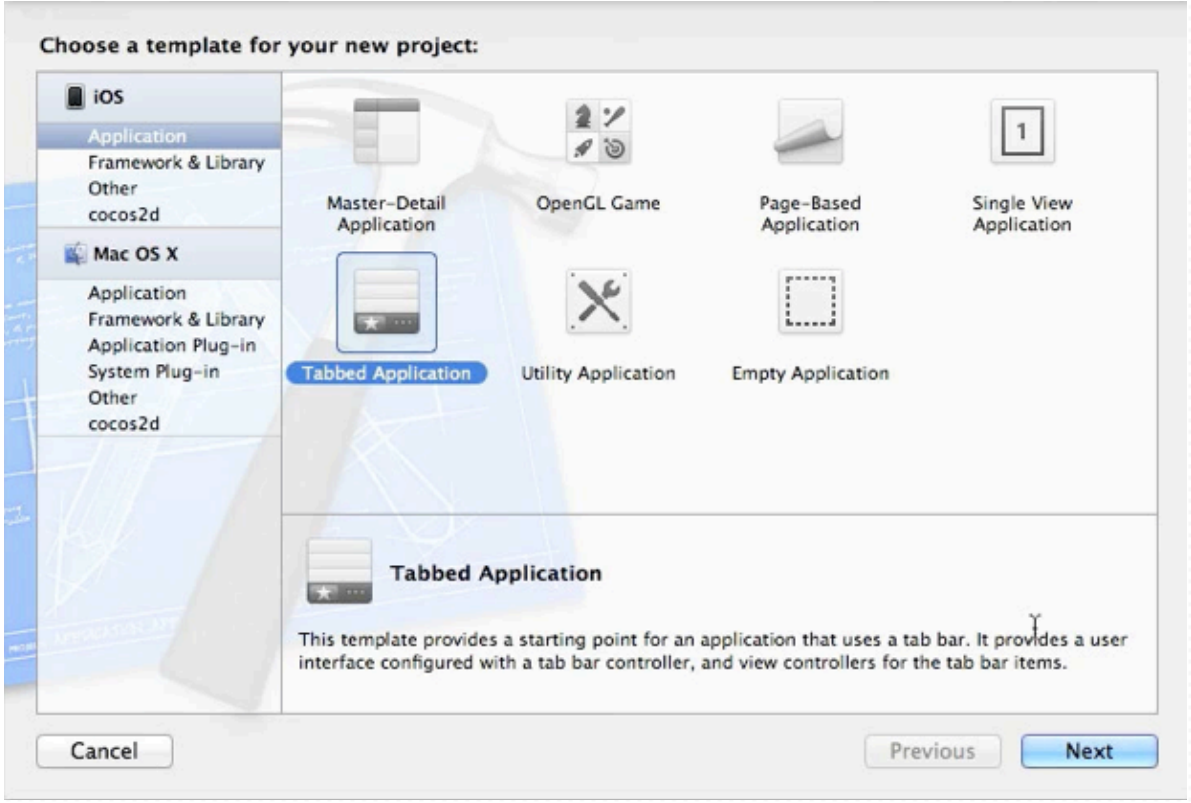
Main navigation between views

Usage of Tab Bar

Non-standard tab bar

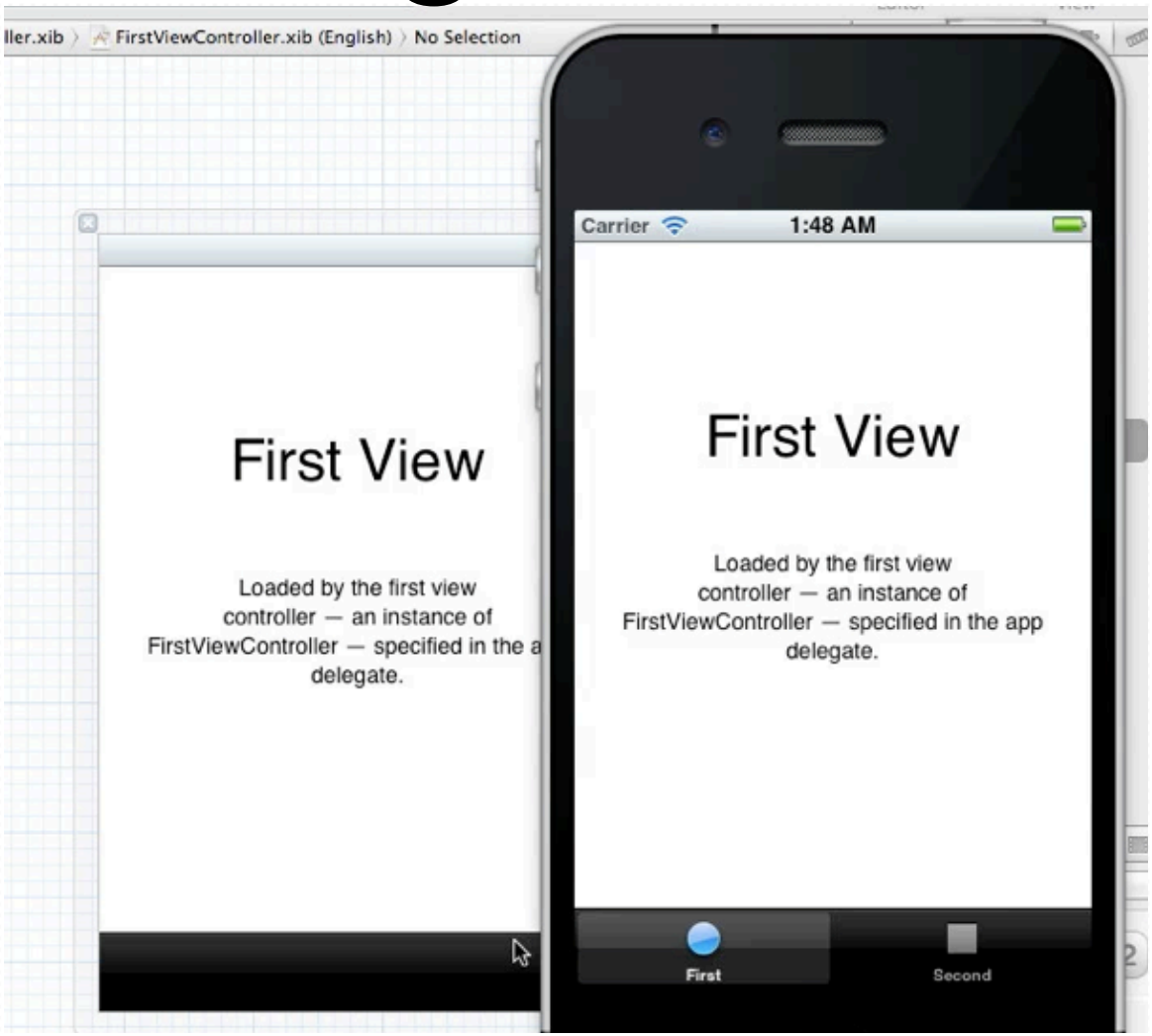
Create our Tabbed Application

Creating a Tabbed App



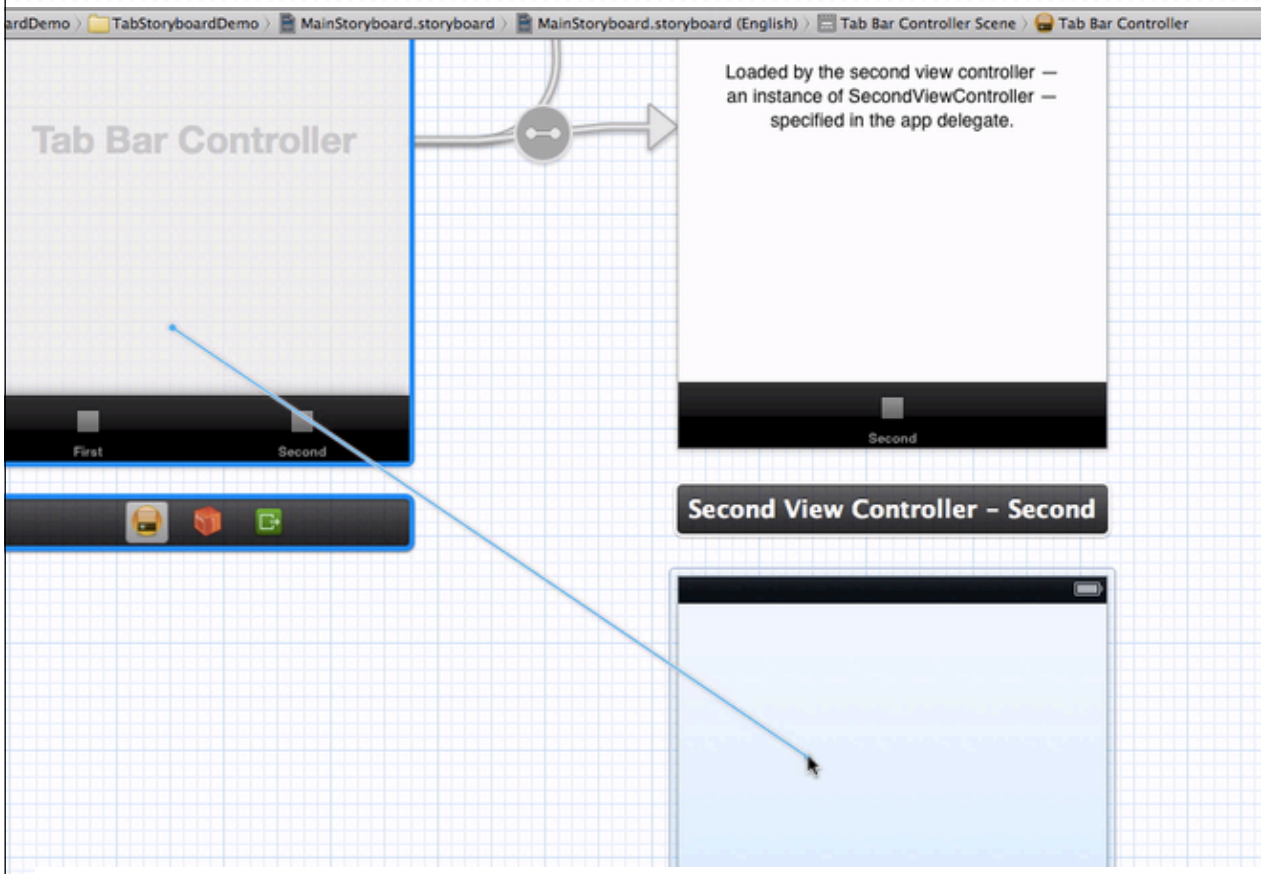
Create a new project with "Tabbed Application" preset.

Creating a Tabbed App



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

The Storyboard way



In storyboard, we right click and drag from the Tab Bar Controller to the new view,
and choose **Relationship Segue**.

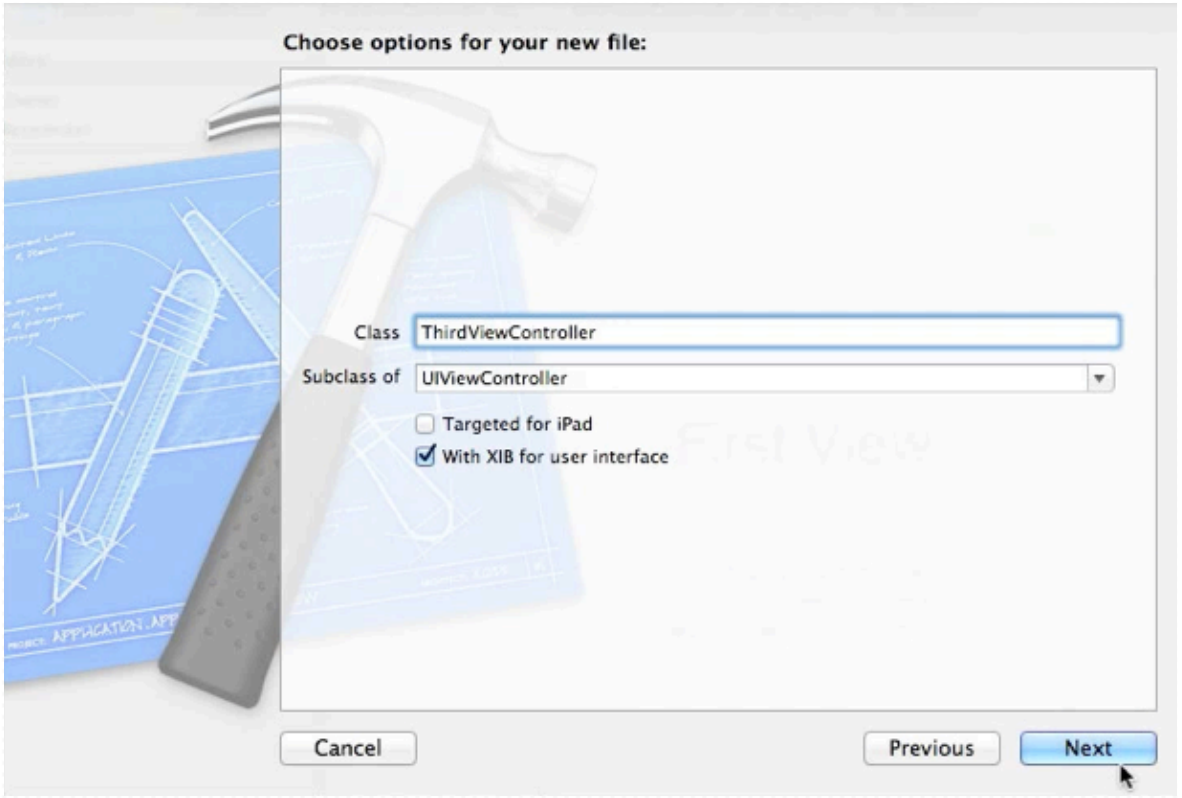
The Program Way

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];
    UIViewController *viewController2 =  [[SecondViewController alloc]
        initWithNibName:@"SecondViewController" bundle:nil];
    self.tabBarController =  [[UITabBarController alloc] init];
    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
@synth
@synthesize tabBarController = _tabBarController;
```

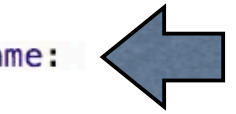
```
ThirdViewController.h
ThirdViewController.m
ThirdViewController.x…
```

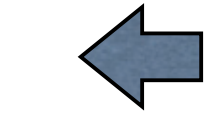
Import our new class into AppDelegate.m

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];
    UIViewController *viewController2 =  [[SecondViewController alloc] initWithNibName:
        @"SecondViewController" bundle:nil];

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

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

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

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

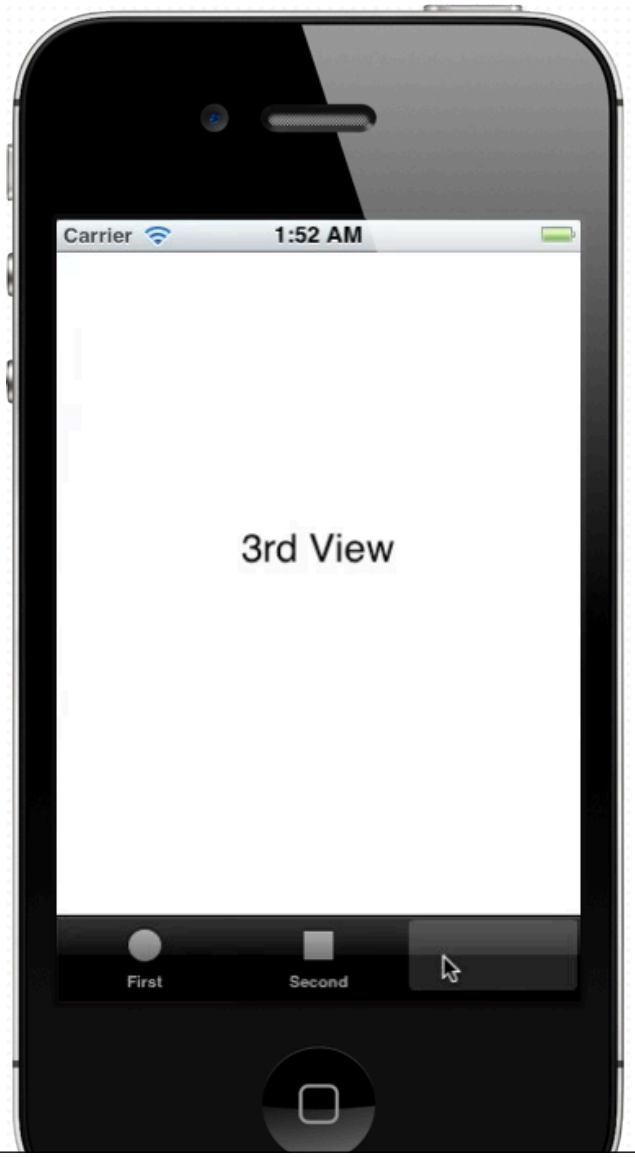




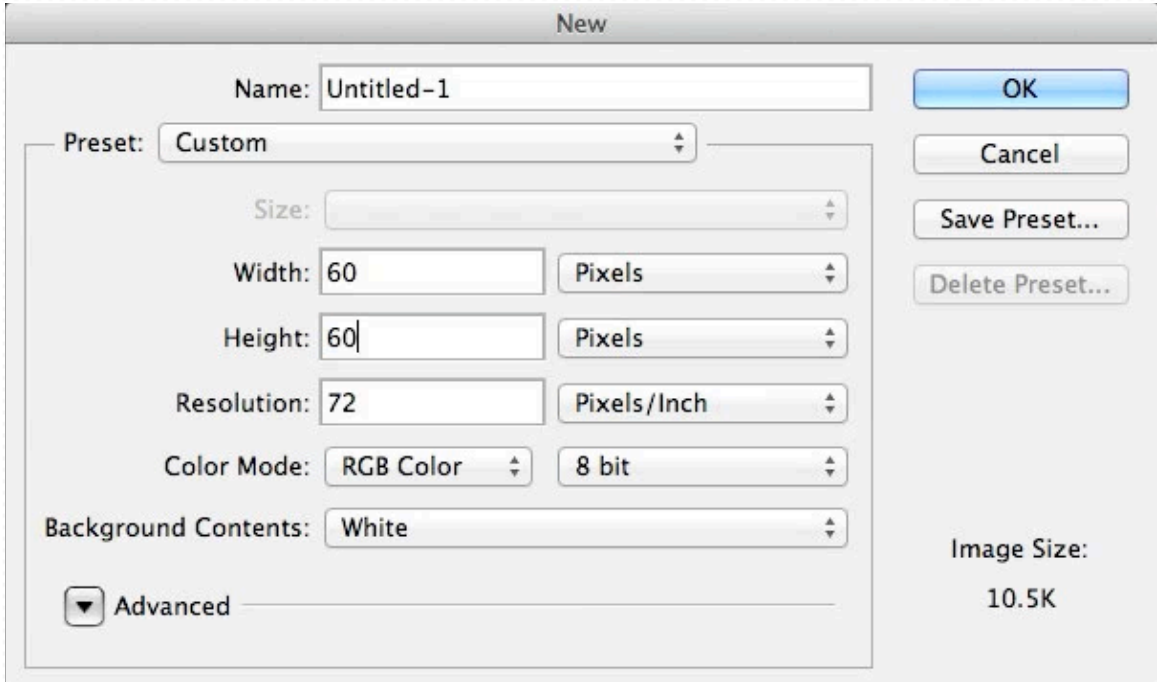
Add the ThirdViewController instance to TabBarController

Creating a Tabbed App

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

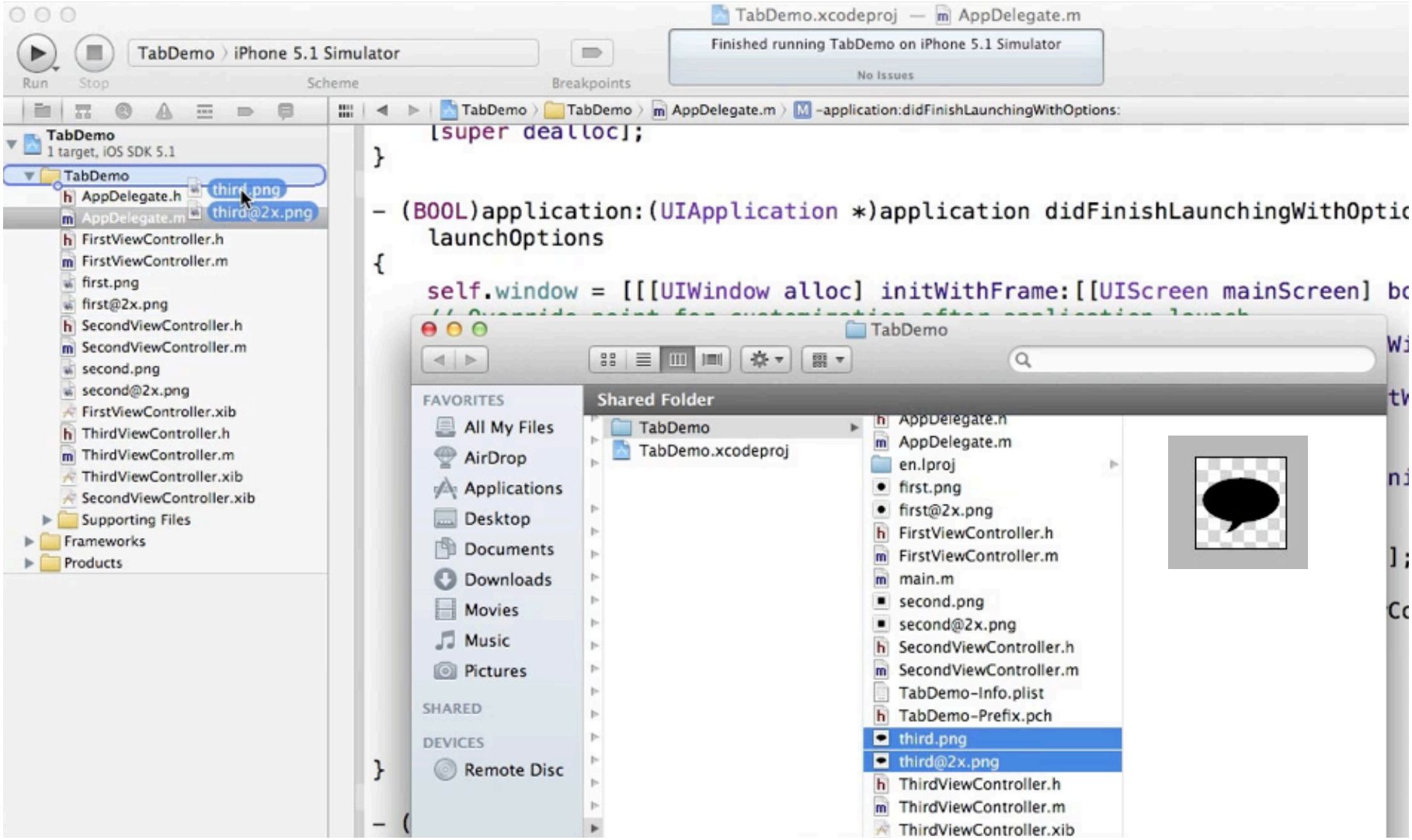


Tab Bar Icon



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

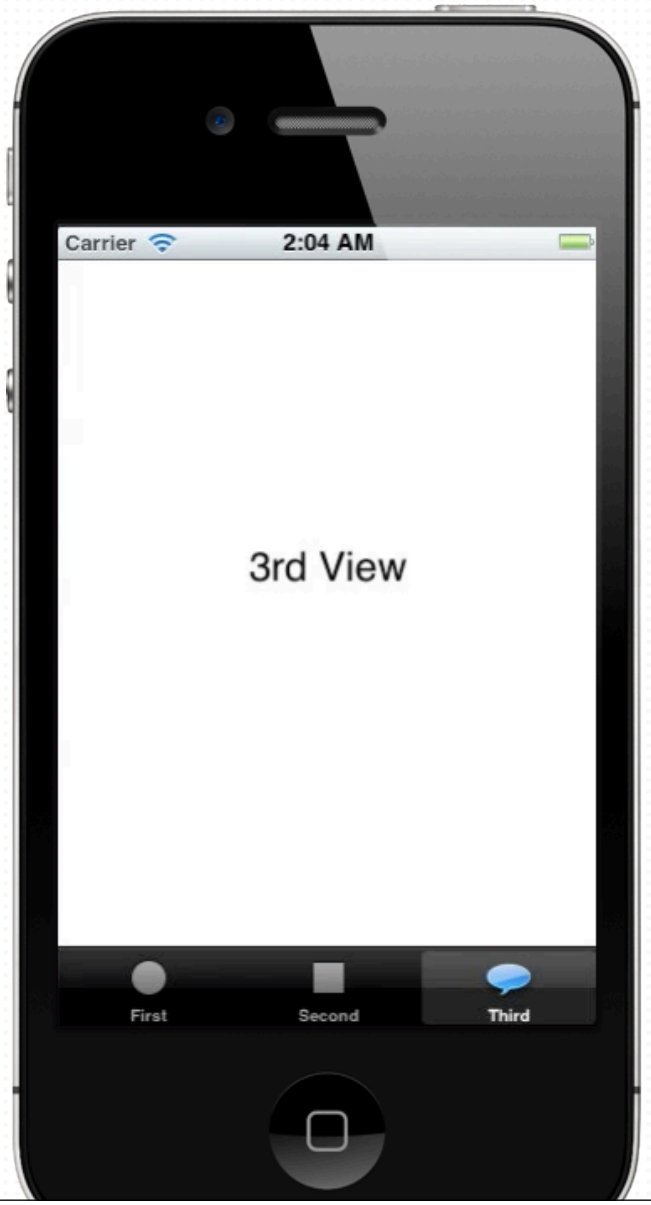
ThirdViewController.m

```
- (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;
}
```

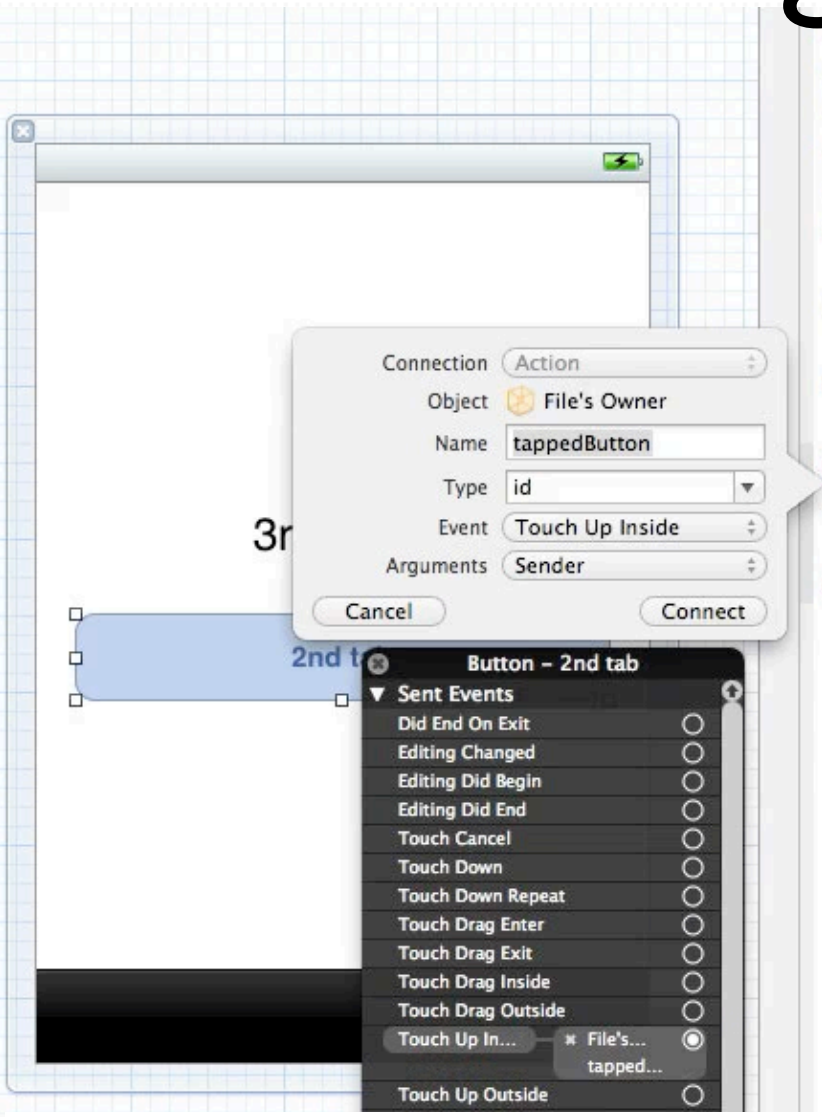
When we init the tabbed view controller, we set the title and image of tabBarItem for the tab.

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

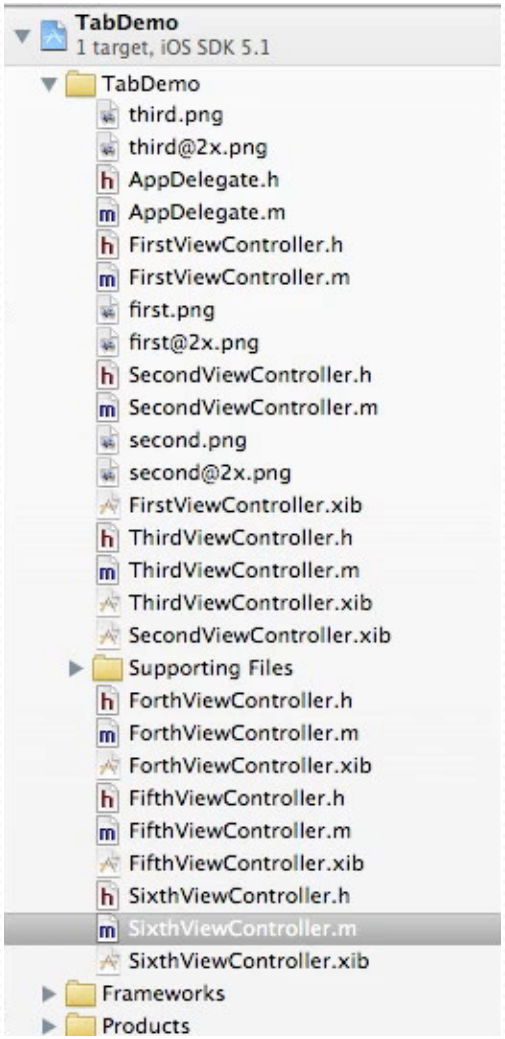
```
- (IBAction)tappedButton:(id)sender {
    self.tabBarController.selectedIndex = 1;
}
```

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.

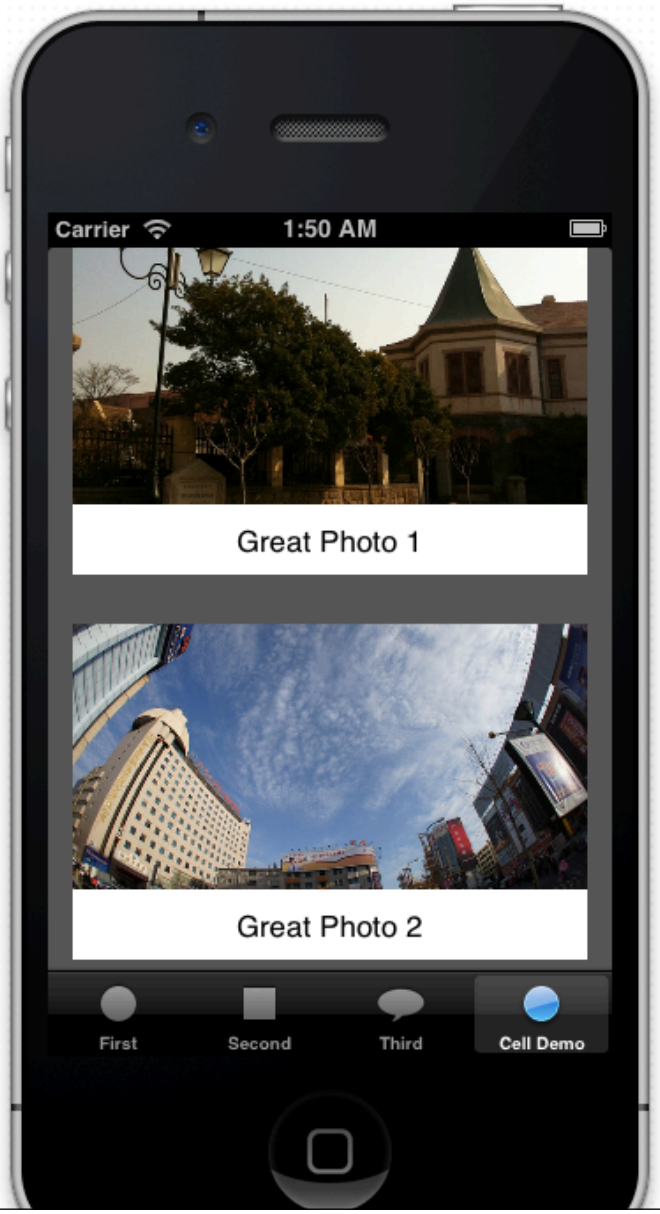
User can decide which 4 to show at the bottom and the order of them.

Custom Table Cell

Let's talk more on table view

Custom Table Cell

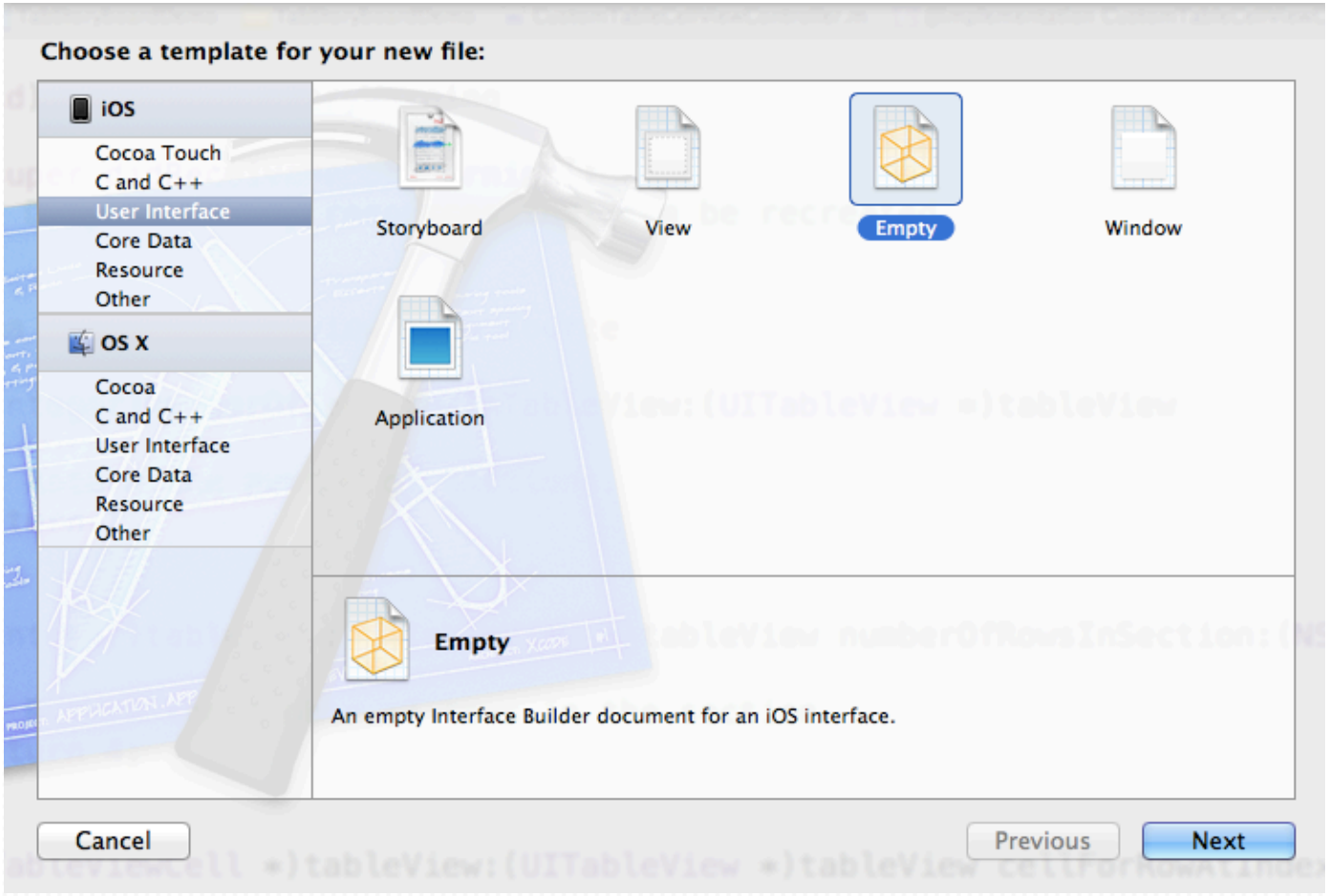
We are going to display some images and meta in table view.



Custom Table Cell

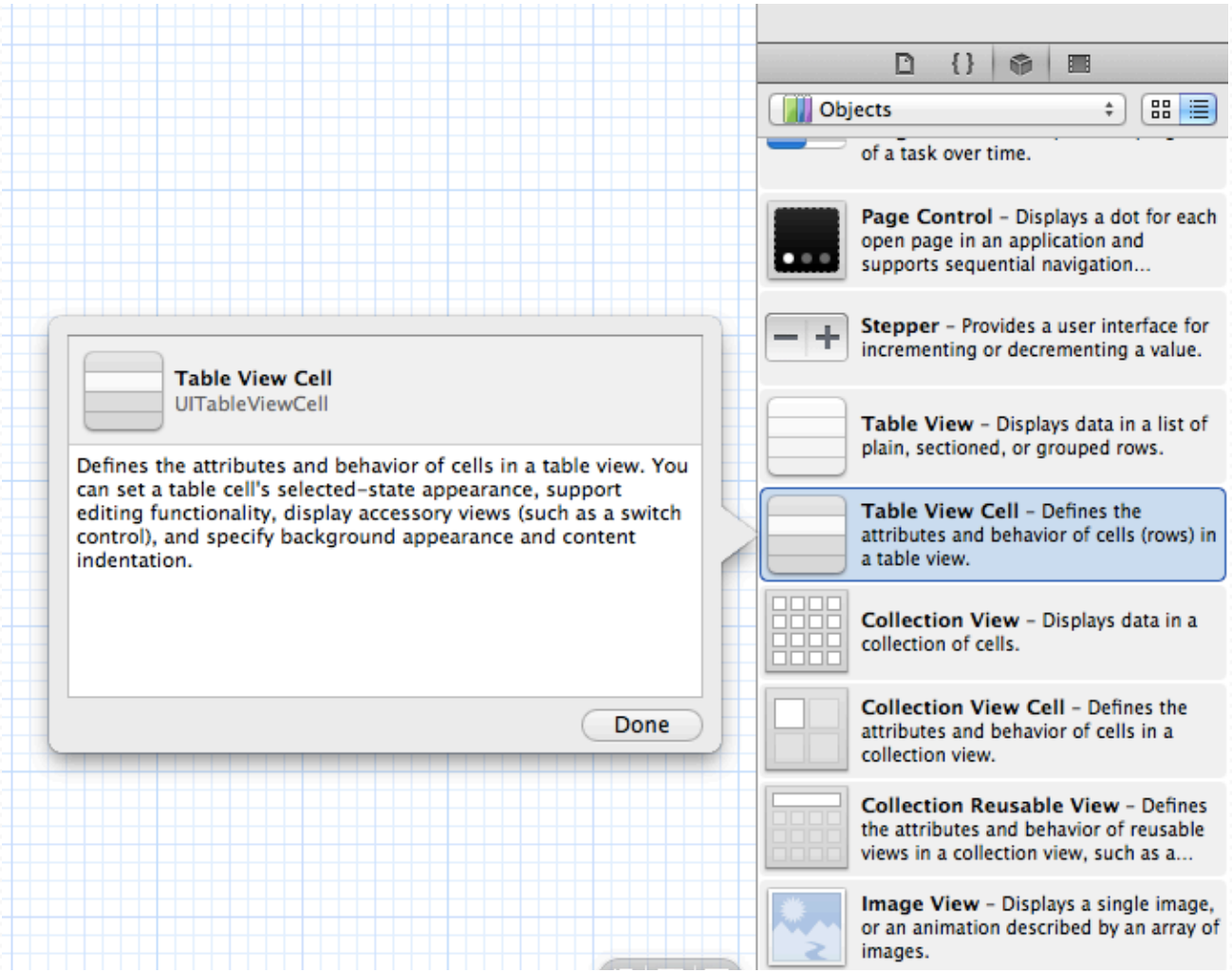
A custom cell view with UIImageView and UILabel

Custom Table Cell



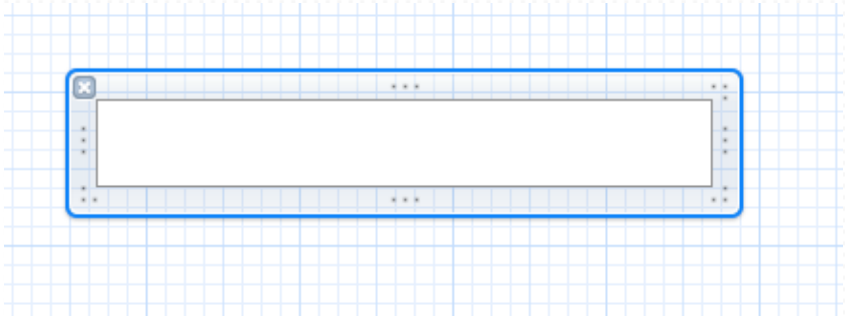
Step 1: New file with empty interface.

Custom Table Cell



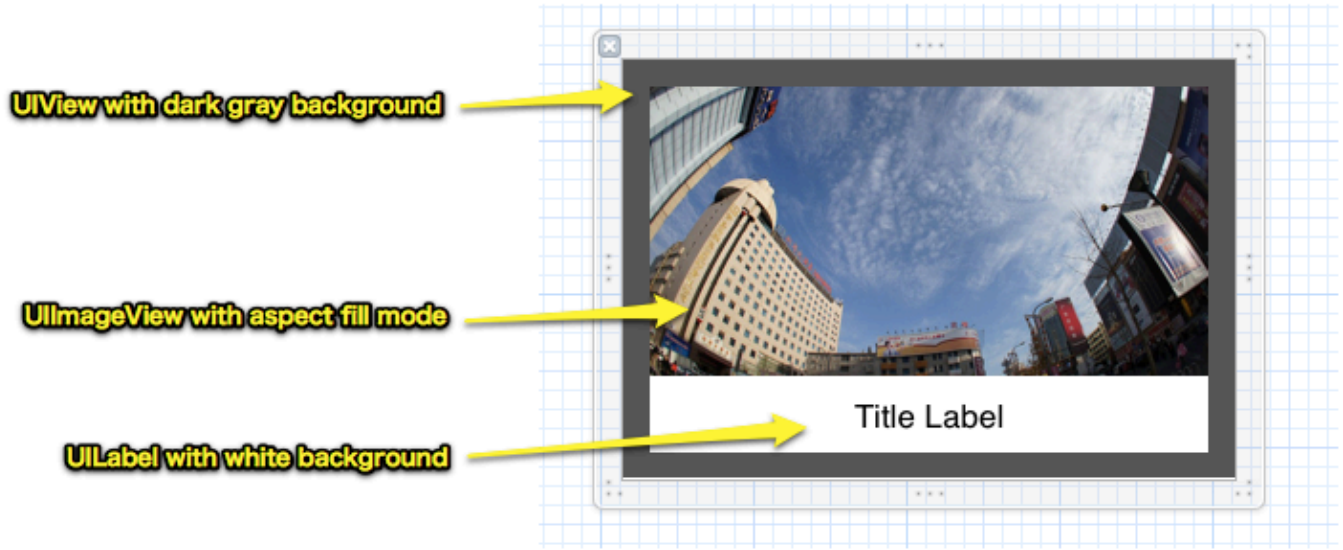
Step 2: Find the Table View Cell component.

Custom Table Cell



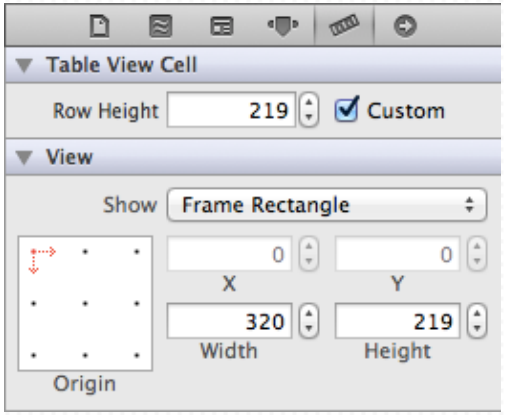
Step 3: Drag it to interface workspace.

Custom Table Cell



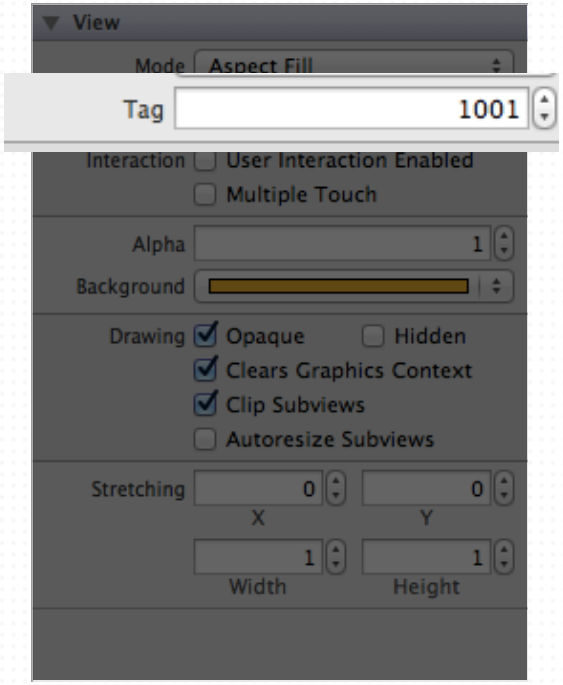
Step 4: Prepare our cell

Custom Table Cell



Step 5: Remember the height of the cell

Custom Table Cell



Step 6: Set a custom integer tag
for both UIImageView and UILabel

Custom Table Cell

[uiview viewWithTag: 123];

viewWithTag allows us to get a view by the given tag integer.
We can avoid IBOutlet linking and still reference to our view.

Custom Table Cell

CustomTableCellViewController.m

```
static NSString *CellIdentifier = @"CustomCell";
```

```
    // register the custom cell
    [self.tableView registerNib:[UINib nibWithNibName:@"CustomCell" bundle:nil] forCellReuseIdentifier:
        CellIdentifier];
```

First, we need to associate the custom cell with a Reusable Cell Identifier

Custom Table Cell

CustomTableCellViewController.m

```
- (CGFloat)tableView:(UITableView *)tableView heightForRowAtIndexPath:(NSIndexPath *)indexPath
{
    return 219.0f;
}
```

Second, we specific the height of the table cell in code.

Custom Table Cell

CustomTableCellViewController.m

```
#define CellImageTag    1001
#define CellLabelTag    1002
```

```
- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath
{
    UITableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:CellIdentifier];

    PhotoRecord *photoRecord = [self.dataArray objectAtIndex:indexPath.row];

    UIImageView *imgView = (UIImageView*)[cell viewWithTag:CellImageTag];
    imgView.image = [UIImage imageNamed:photoRecord.fileName];

    UILabel *titleLabel = (UILabel*)[cell viewWithTag:CellLabelTag];
    titleLabel.text = photoRecord.title;


    return cell;
}
```

Next, we construct the cell in code.

Custom Table Cell

PhotoRecord.h

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

@interface PhotoRecord : NSObject

@property (nonatomic, strong) NSString *fileName;
@property (nonatomic, strong) NSString *title;

@end
```

Let's prepare a class named **PhotoRecord** to store the data of each photo record.

Custom Table Cell

CustomTableCellViewController.m

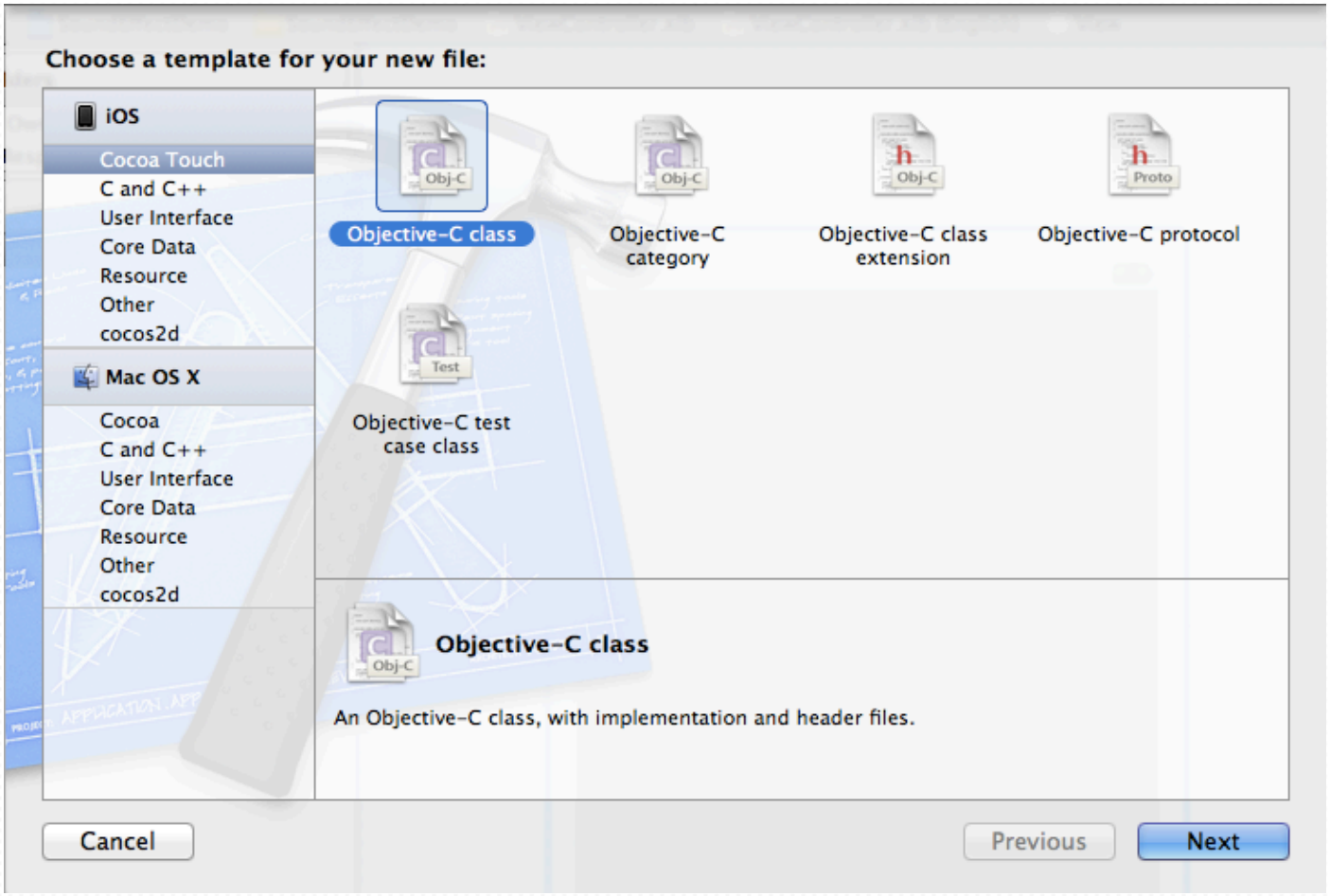
```
// prepare the data
PhotoRecord *record1 = [[PhotoRecord alloc] init];
record1.fileName = @"photo1.png";
record1.title = @"Great Photo 1";

PhotoRecord *record2 = [[PhotoRecord alloc] init];
record2.fileName = @"photo2.png";
record2.title = @"Great Photo 2";

PhotoRecord *record3 = [[PhotoRecord alloc] init];
record3.fileName = @"photo3.png";
record3.title = @"Great Photo 3";

PhotoRecord *record4 = [[PhotoRecord alloc] init];
record4.fileName = @"photo4.png";
record4.title = @"Great Photo 4";

self.dataArray = @[record1, record2, record3, record4];
```

And we prepare the data when view loads

Playing Sound Effects

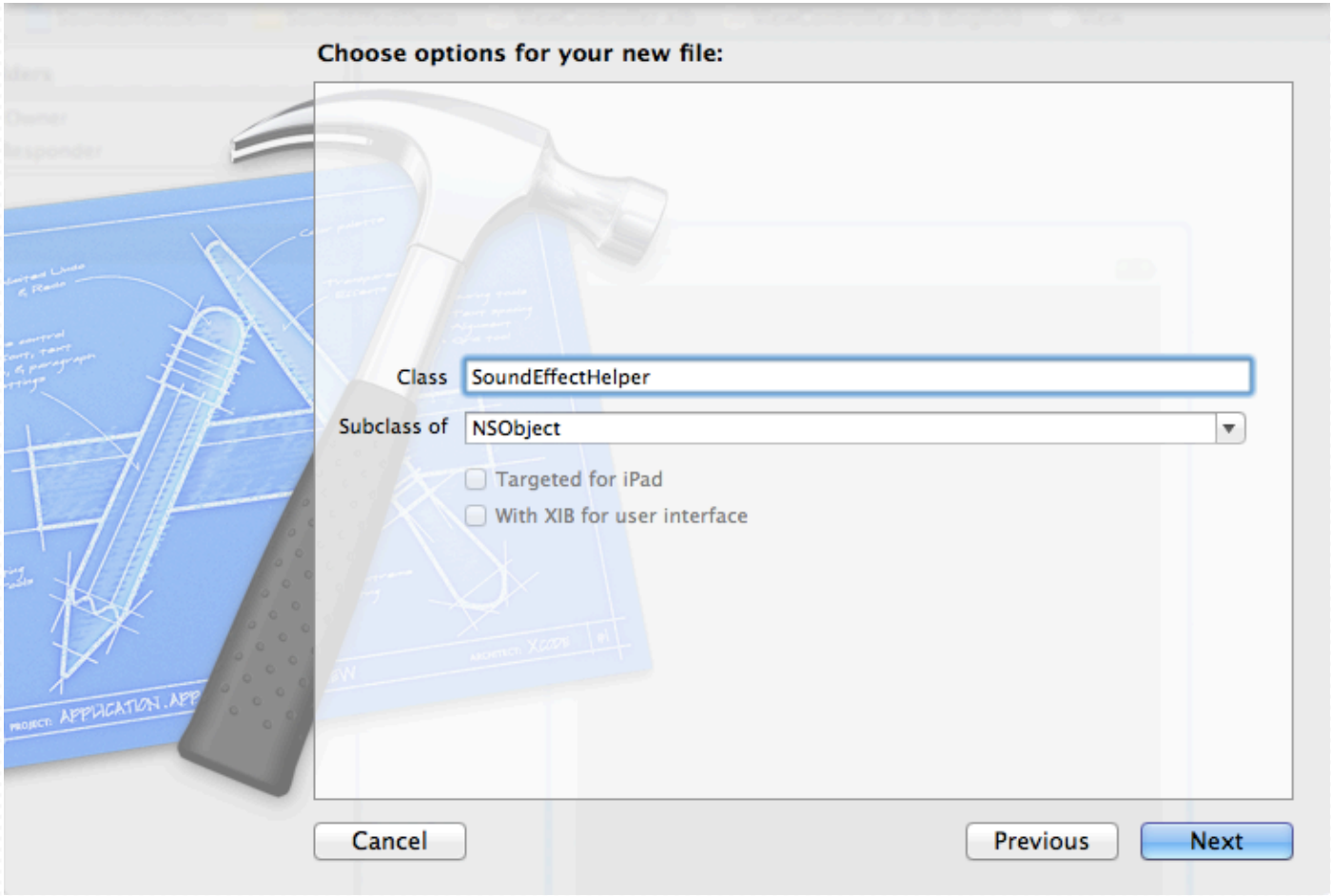
We will explore two approaches to play sounds

Playing Sound Effects



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

Playing Sound Effects

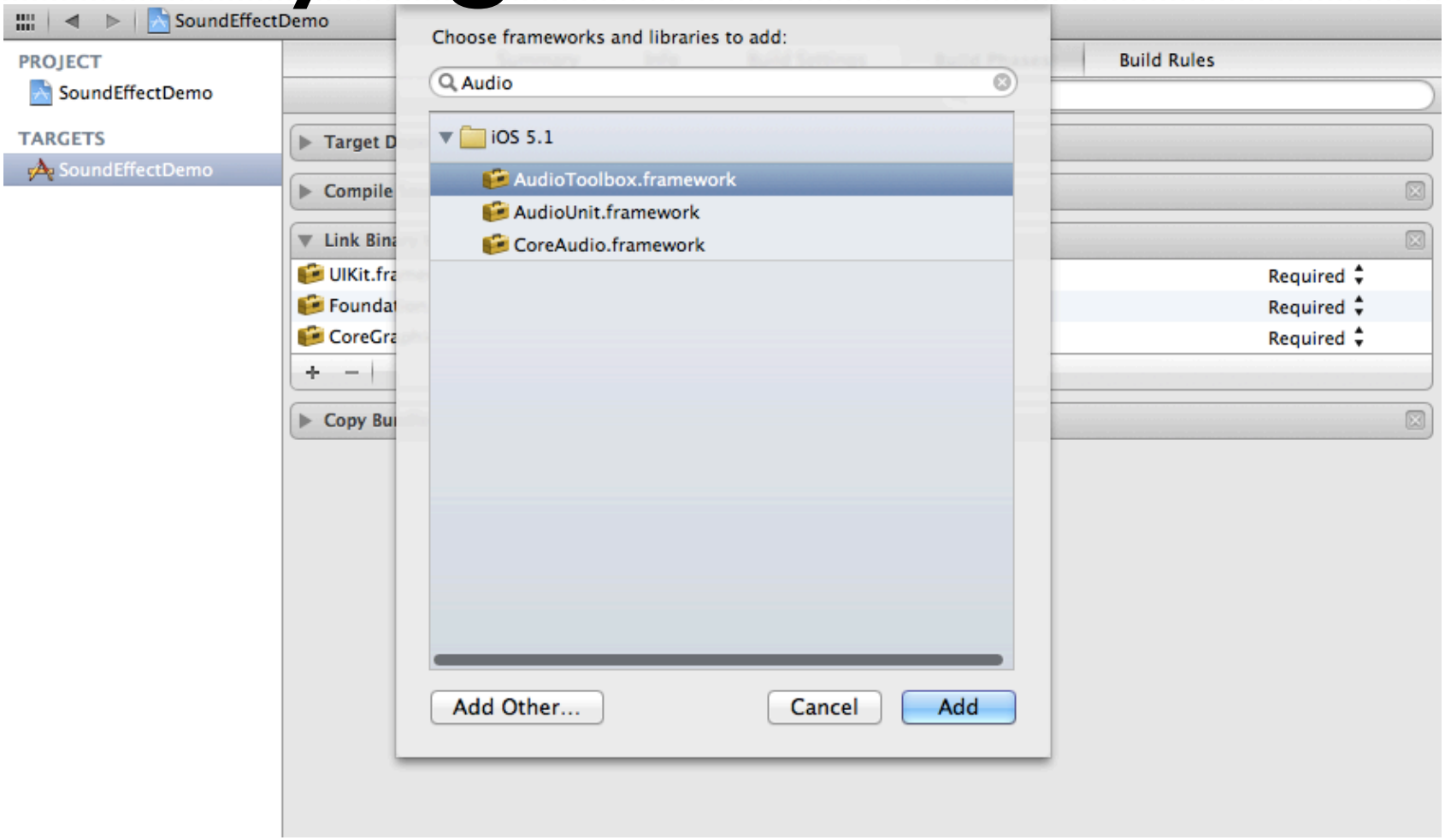


Give the class a name and set it to be subclass of NSObject.

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.

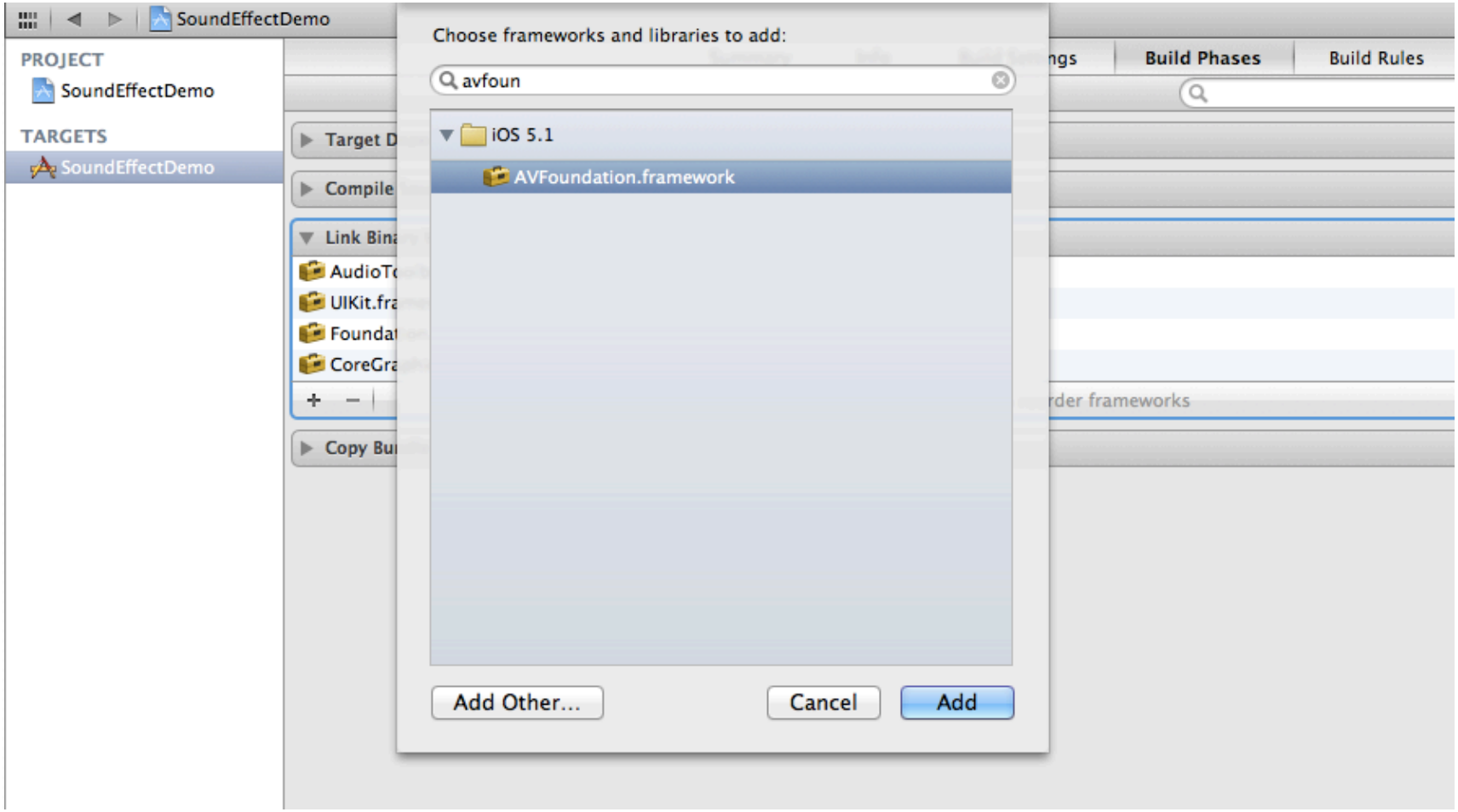
Playing Sound Effects

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.

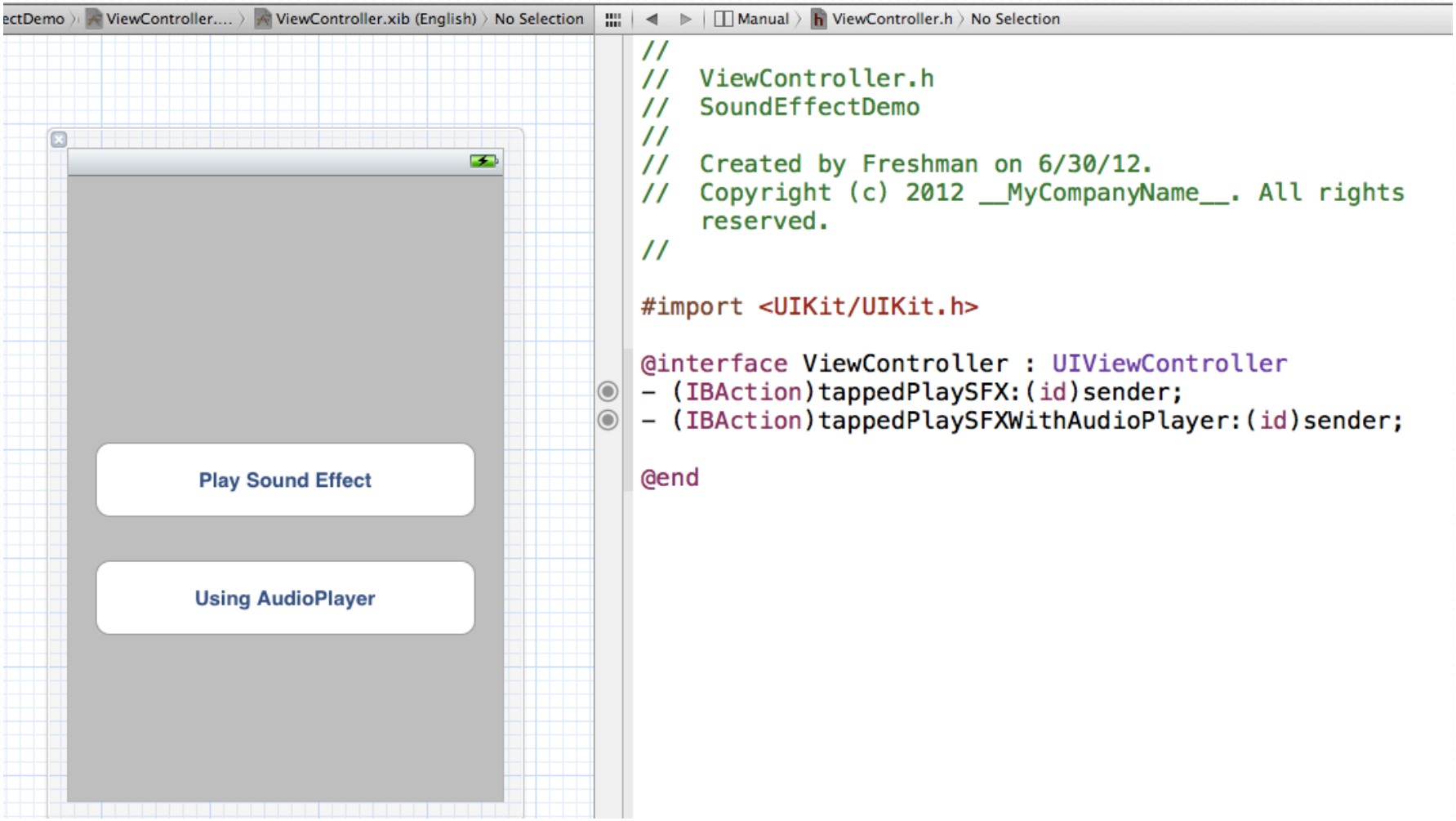
Playing Sound Effects

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 30 seconds to talk about your project ideas