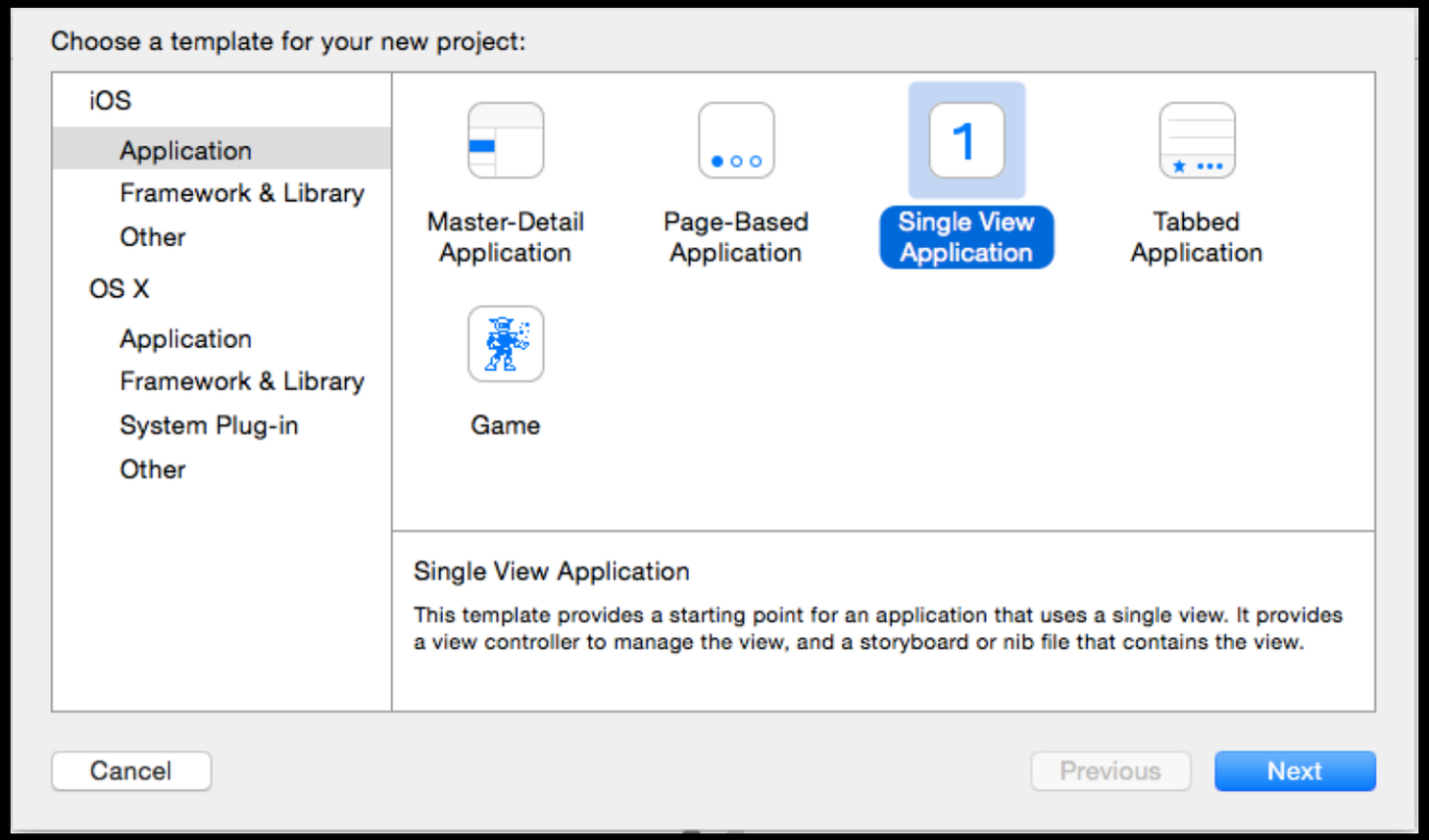
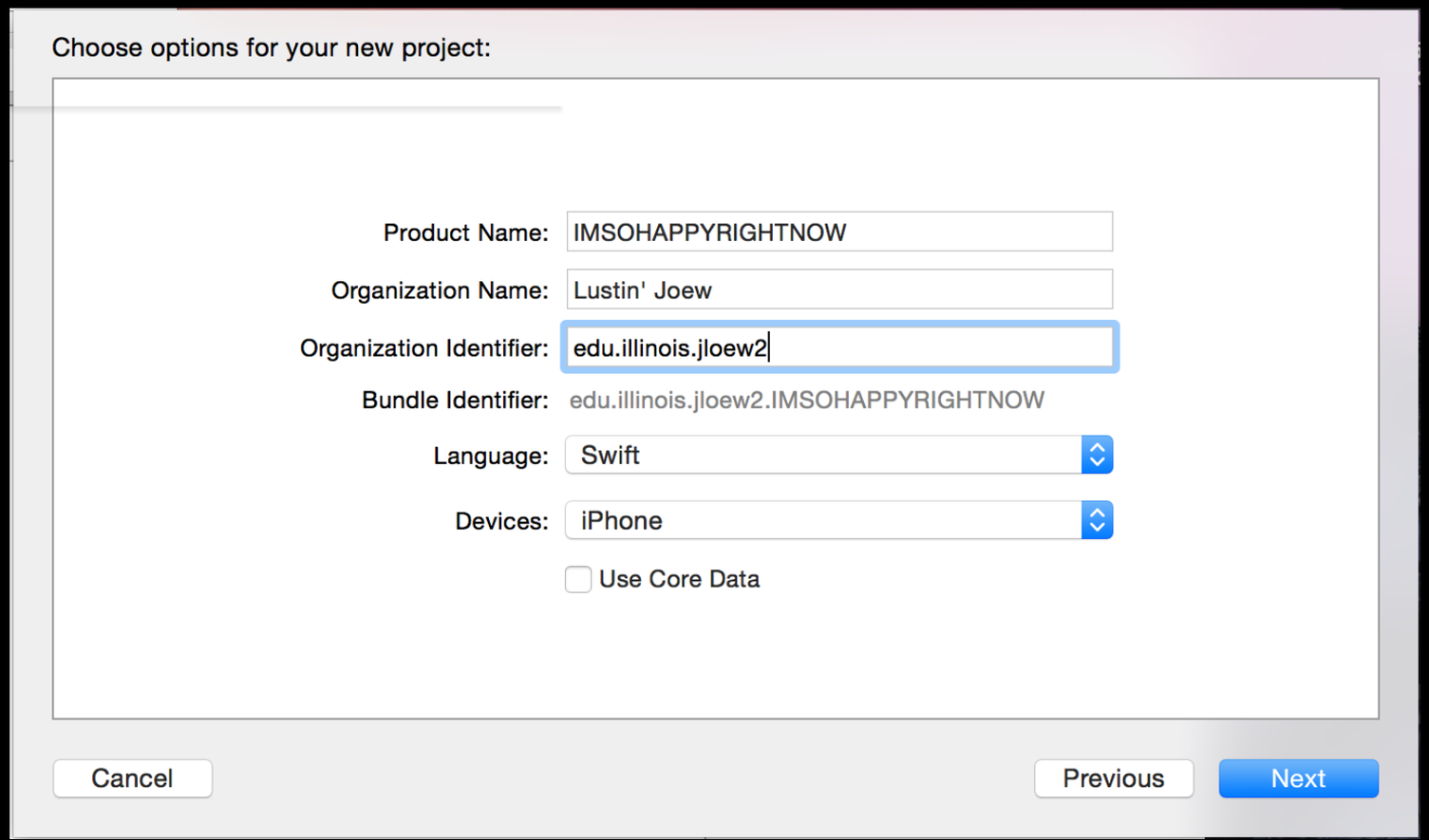


First, let's create a new Xcode project. A Single View Application. Language swift. Targeted toward iPhone.



Xcode File Edit View Find Navigate Editor Product Debug Source Control Win

New ▶

Add Files to “IMSOHAPPYRIGHTNOW”… ⌥⌘A

Open… ⌘O
Open Recent ▶
Open Quickly… ⇧⌘O

Close Window ⌘W
Close Tab
Close “Main.storyboard” ^⌘W
Close Project ⌥⌘W

Save ⌘S
Duplicate… ⇧⌘S
Revert to Saved

Tab ⌘T
Window ⇧⌘T

File… ⌘N
Playground… ⌥⇧⌘N
Target…
Project… ⇧⌘N
Workspace… ^⌘N

Group ⌥⌘N
Group from Selection

IMSOHAPPY
targets, iOS
IMSOHAP
AppDel
ViewCo
Main.st
Images
Launch
Suppor
IMSOHAP
Products

RIGHTN
toryboa

Choose a template for your new file:

- iOS
  - Source
  - User Interface
  - Core Data
  - Resource
  - Other
- OS X
  - Source
  - User Interface
  - Core Data
  - Resource
  - Other

Cocoa Class

Test Case Class

Playground

Swift File

Objective-C File

Header File

C File

C++ File

Swift File

An empty Swift file.

Cancel

Previous

Next

Save As: FaceView.swift

Tags:

IMSOHAPPYRIGHTNOW

Search

Devices
- rmbp
- SSD256
- Remote Disc

Favorites
- Recents
- Dropbox
- iCloud Drive
- Applications
- Desktop
- Documents
- Downloads
- jloloew

Shared
- imac

IMSOHAPPYRIGHTNOW
IMSOHAPPY…W.xcodeproj
IMSOHAPPY…TNOWTests

AppDelegate.swift
Base.lproj
Images.xcassets
Info.plist
ViewController.swift

Group: IMSOHAPPYRIGHTNOW

Targets:
- IMSOHAPPYRIGHTNOW
- IMSOHAPPYRIGHTNOWTests

New Folder

Cancel

Create

```swift
//
//  FaceView.swift
//  IMSOHAPPYRIGHTNOW
//
//  Created by Justin Loew on 10/4/14.
//  Copyright (c) 2014 Lustin' Joew. All rights reserved.
//

import Foundation


import UIKit


class FaceView: UIView {

    var scale: Float = 0.90 {
        // didSet is called every time scale is set (after it has the new value)
        didSet {
            // don't allow zero scale
            if scale == 0 {
                scale = Float(0.90)
            }
            // any time our scale changes, call for redraw
            setNeedsDisplay()
        }
    }

}
```

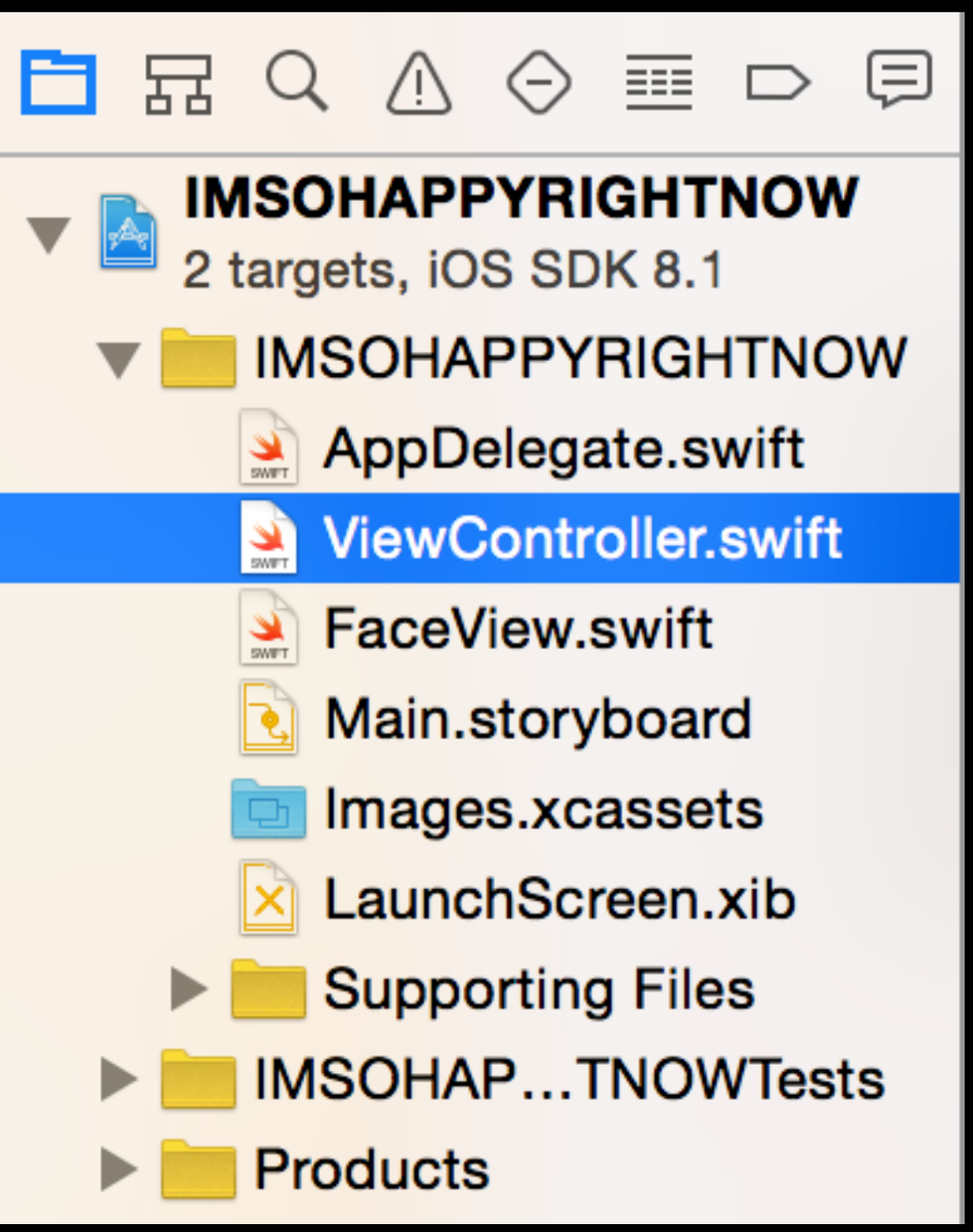
Add a '`scale`' property to our `FaceView` class. Make sure `FaceView` inherits from `UIView` (`FaceView: UIView`) and you import UIKit.

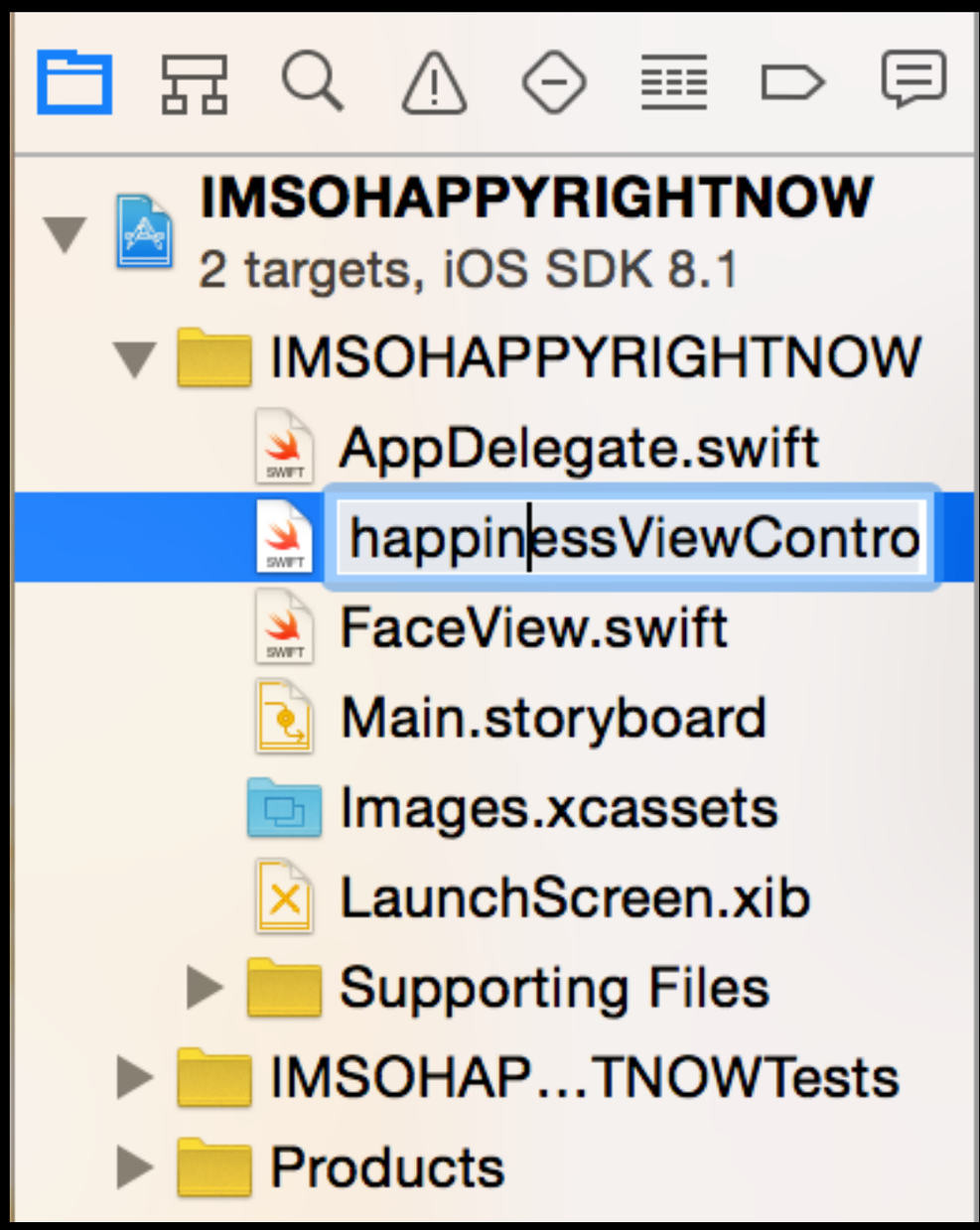
```swift
//
//  FaceView.swift
//  IMSOHAPPYRIGHTNOW
//
//  Created by Justin Loew on 10/4/14.
//  Copyright (c) 2014 Lustin' Joew. All rights reserved.
//

import Foundation


import UIKit


class FaceView: UIView {

    var scale: Float = 0.90 {
        // didSet is called every time scale is set (after it has the new value)
        didSet {
            // don't allow zero scale
            if scale == 0 {
                scale = Float(0.90)
            }
            // any time our scale changes, call for redraw
            setNeedsDisplay()
        }
    }

    var dataSource: FaceViewDataSource? = nil


    override init(frame: CGRect) {
        super.init(frame: frame)
    }

    required init(coder aDecoder: NSCoder) {
        super.init(coder: aDecoder)
    }


    func drawCircleAtPoint(p: CGPoint, withRadius r: CGFloat, inContext context: CGContextRef) {
        UIGraphicsPushContext(context)

        CGContextBeginPath(context)
        CGContextAddArc(context, p.x, p.y, r, CGFloat(0.0), CGFloat(2*M_PI), 1)
        CGContextStrokePath(context)

        UIGraphicsPopContext()
    }

    func pinch(gesture: UIPinchGestureRecognizer) {
        if (gesture.state == .Changed) || (gesture.state == .Ended) {
            scale *= Float(gesture.scale)   // adjust our scale
            gesture.scale = 1   // reset gestures scale to 1 (so future changes are incremental, not cumulative)
        }
    }
}


protocol FaceViewDataSource {
    func smileForFaceView(sender: FaceView)
}
```

At the bottom, there's a `FaceViewDataSource` protocol. There are a few initializers, a `drawCircleAtPoint` method full of wizardry, and a `pinch` method.

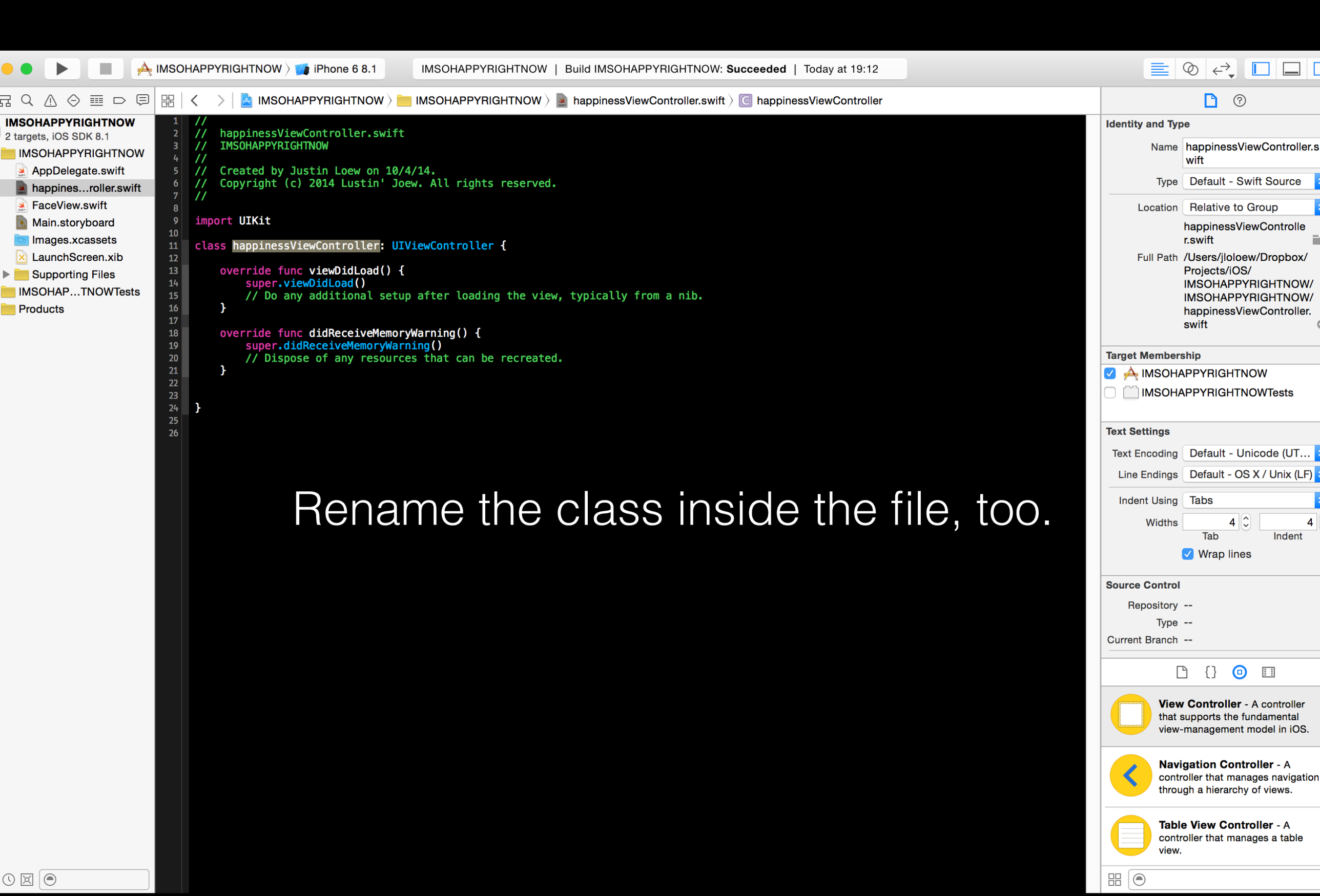
```
}

override func drawRect(rect: CGRect) {
    var context = UIGraphicsGetCurrentContext()

    var midpoint = CGPointMake(self.bounds.origin.x + self.bounds.size.width/2, self.bounds.origin.y + self.bounds.size.height/2)

    var faceSize: CGFloat
    if self.bounds.size.width < self.bounds.size.height {
        faceSize = self.bounds.size.width / 2.0 * self.scale
    } else {
        faceSize = self.bounds.size.height / 2 * self.scale
    }

    CGContextSetLineWidth(context, 5)
    UIColor.blueColor().setStroke()

    drawCircleAtPoint(midpoint, withRadius: faceSize, inContext: context)

    let EYE_H = CGFloat(0.35)
    let EYE_V = CGFloat(0.35)
    let EYE_RADIUS = CGFloat(0.10)

    var eyePoint = CGPointMake(midpoint.x - faceSize * EYE_H, midpoint.y - faceSize * EYE_V)

    drawCircleAtPoint(eyePoint, withRadius: faceSize * EYE_RADIUS, inContext: context)  // left eye
    eyePoint.x += faceSize * EYE_H * 2
    drawCircleAtPoint(eyePoint, withRadius: faceSize * EYE_RADIUS, inContext: context)  // right eye

    let MOUTH_H = CGFloat(0.45)
    let MOUTH_V = CGFloat(0.40)
    let MOUTH_SMILE = CGFloat(0.25)

    var mouthStart = CGPointMake(midpoint.x - MOUTH_H * faceSize, midpoint.y + MOUTH_V * faceSize)
    var mouthEnd: CGPoint = mouthStart
    mouthEnd.x += MOUTH_H * faceSize * 2
    var mouthCP1: CGPoint = mouthStart
    mouthCP1.x += MOUTH_H * faceSize * 2/3
    var  mouthCP2: CGPoint = mouthEnd
    mouthCP2.x -= MOUTH_H * faceSize * 2/3

    var smile = CGFloat(dataSource!.smileForFaceView(self))
    if smile < -1 {
        smile = -1;
    }
    if smile > 1 {
        smile = 1;
    }

    var smileOffset: CGFloat = MOUTH_SMILE * faceSize * smile
    mouthCP1.y += smileOffset
    mouthCP2.y += smileOffset

    CGContextBeginPath(context)
    CGContextMoveToPoint(context, mouthStart.x, mouthStart.y)
    CGContextAddCurveToPoint(context, mouthCP1.x, mouthCP1.y, mouthCP2.x, mouthCP2.y, mouthEnd.x, mouthEnd.y)   // bezier curve
    CGContextStrokePath(context)
  }
}


protocol FaceViewDataSource {
```

Hoo boy, this one's a doozy. It uses **`CoreGraphics`** to to define lines, then draw them in order to draw the face.

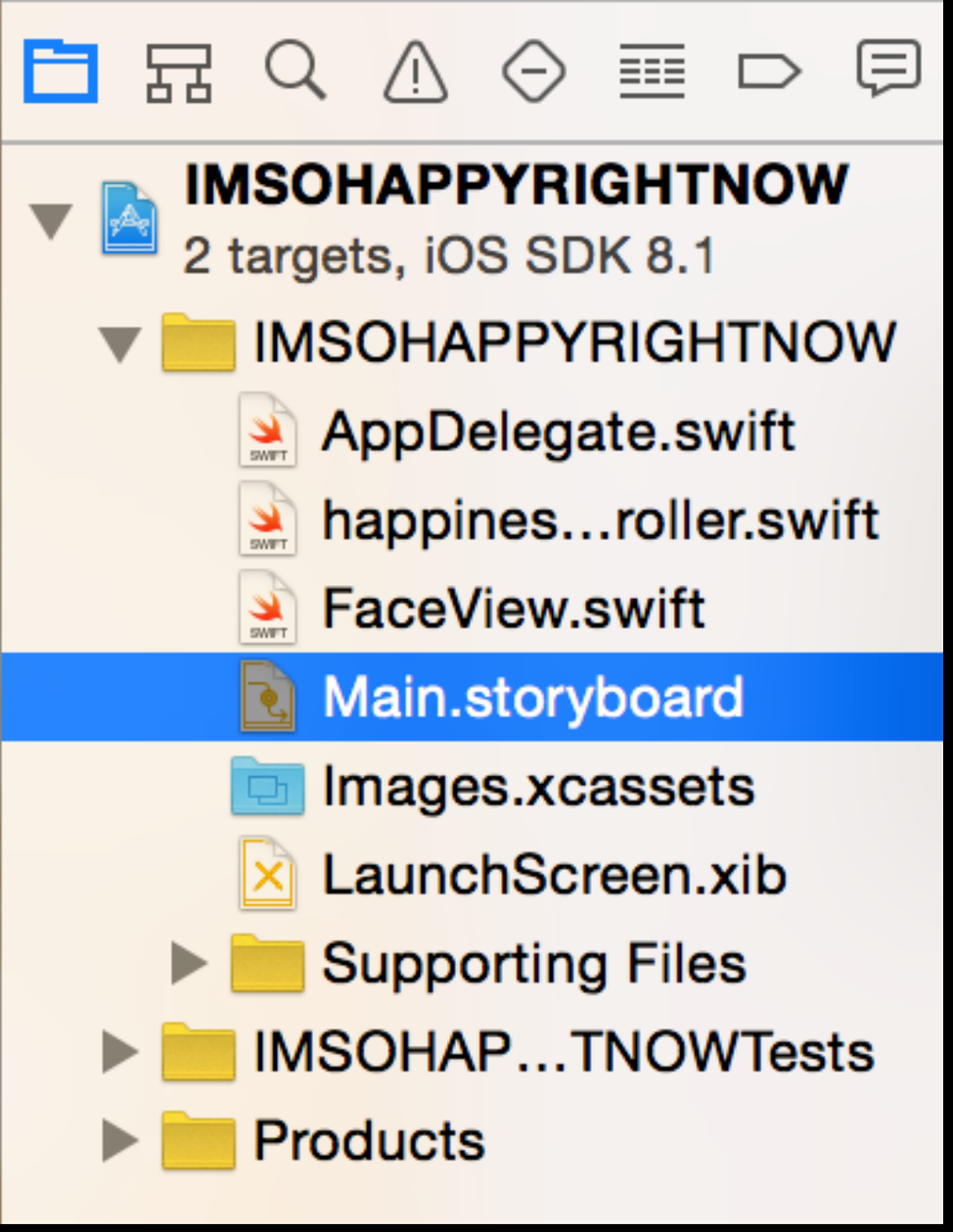




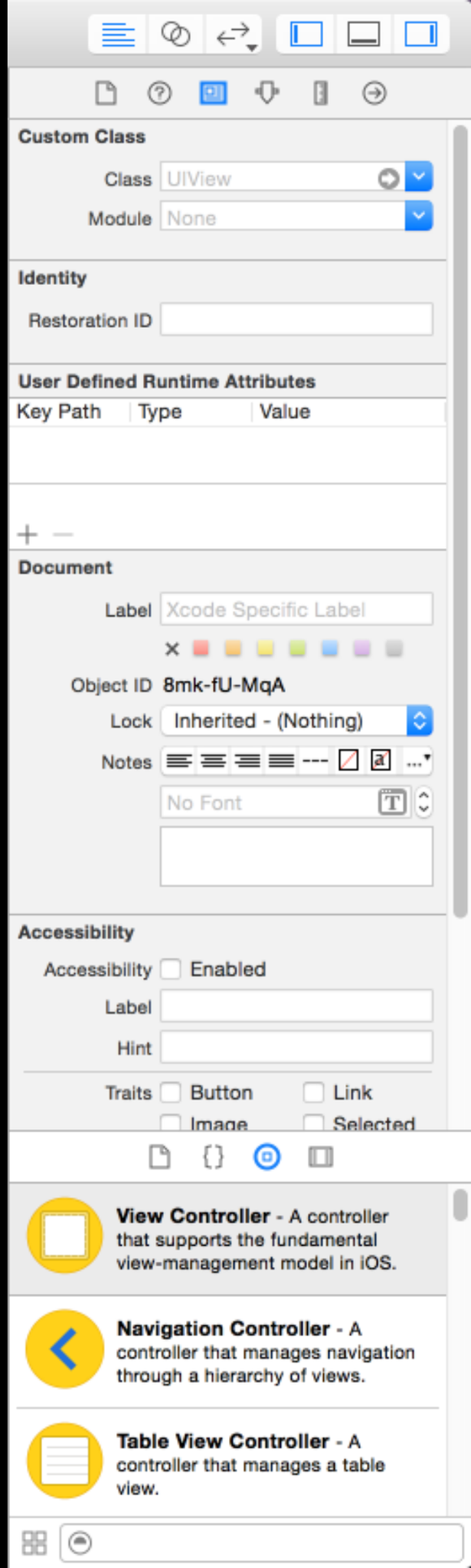
Rename the `ViewController` to `happinessViewController`.



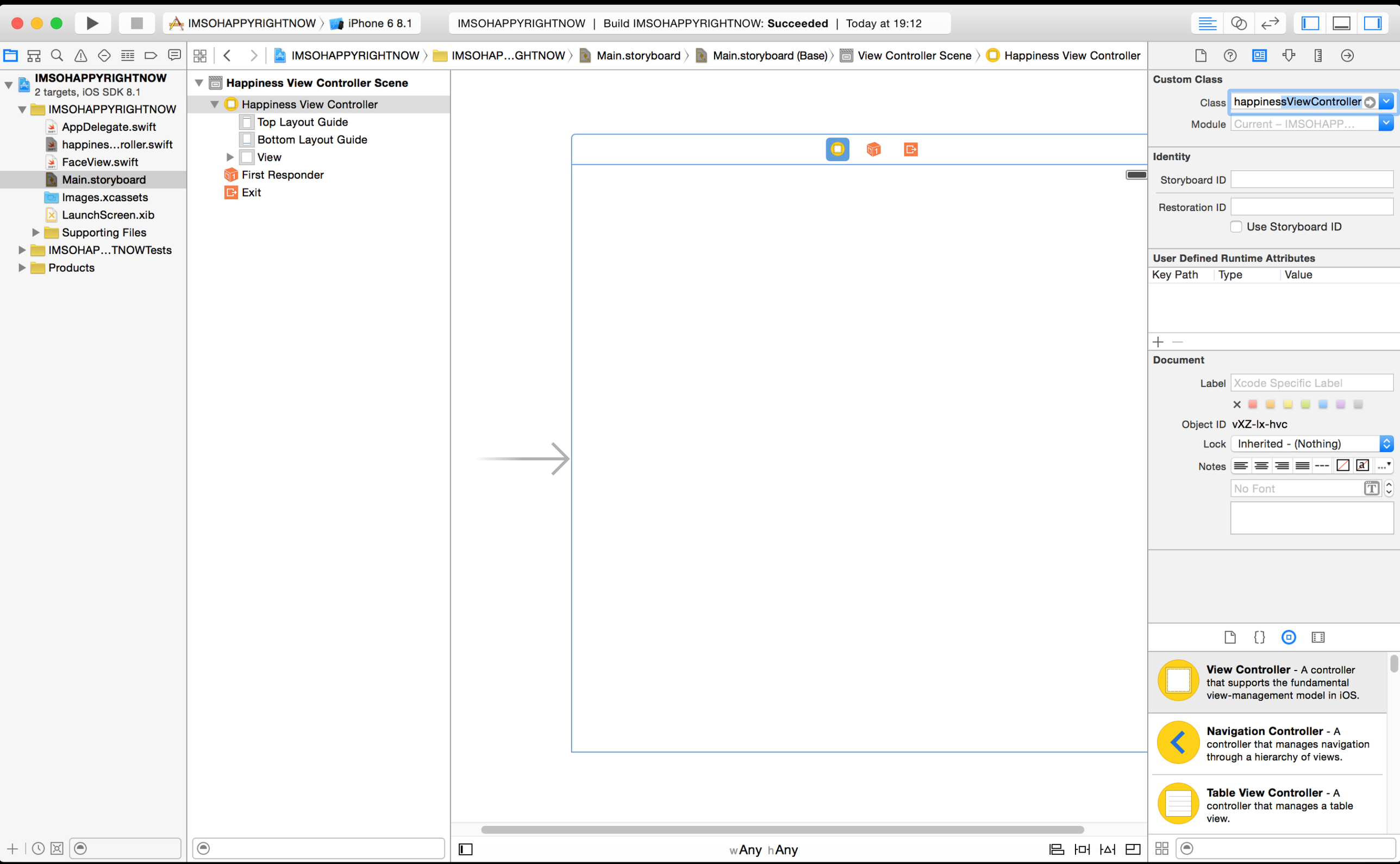
Rename the class inside the file, too.



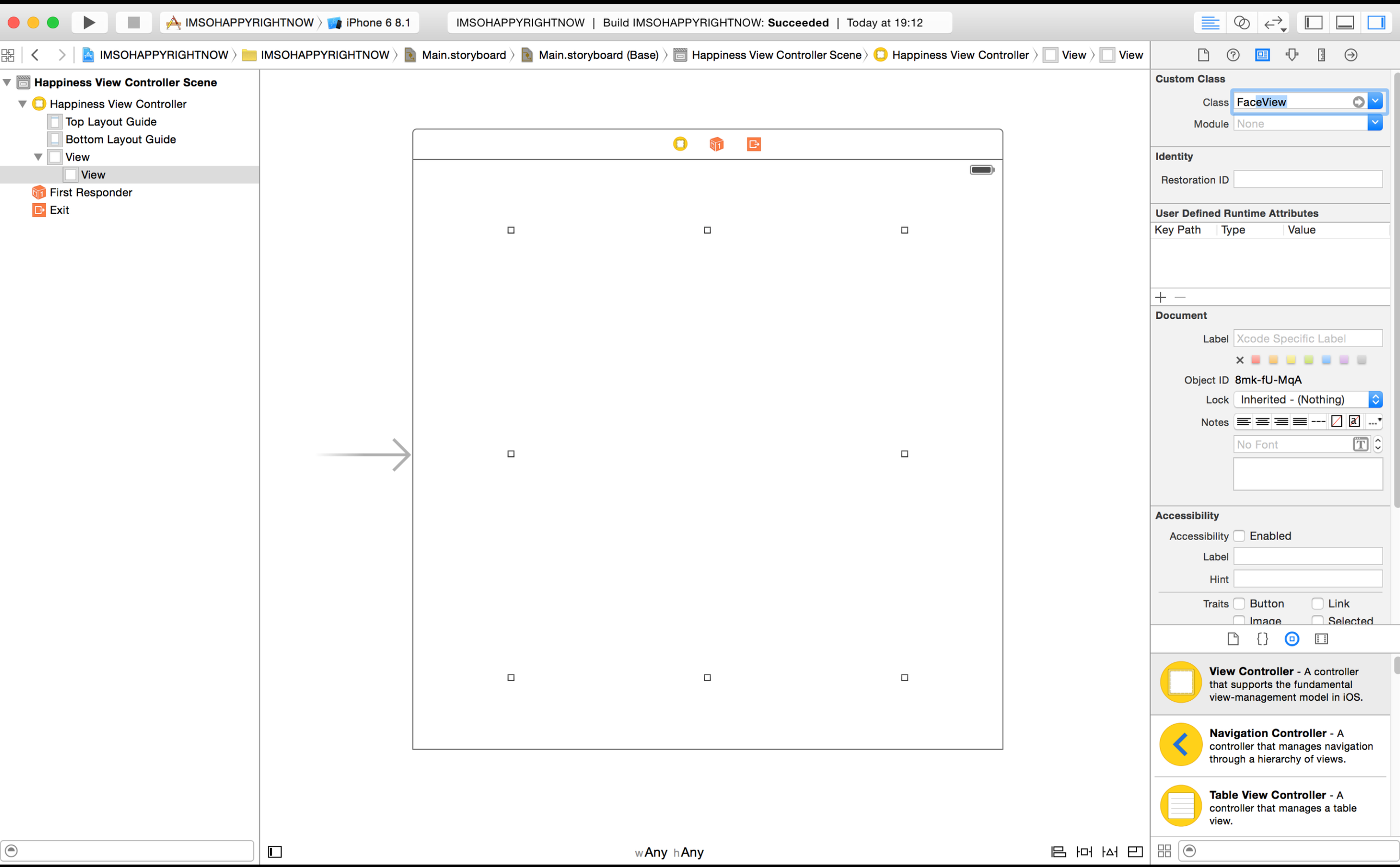
Now open the storyboard.



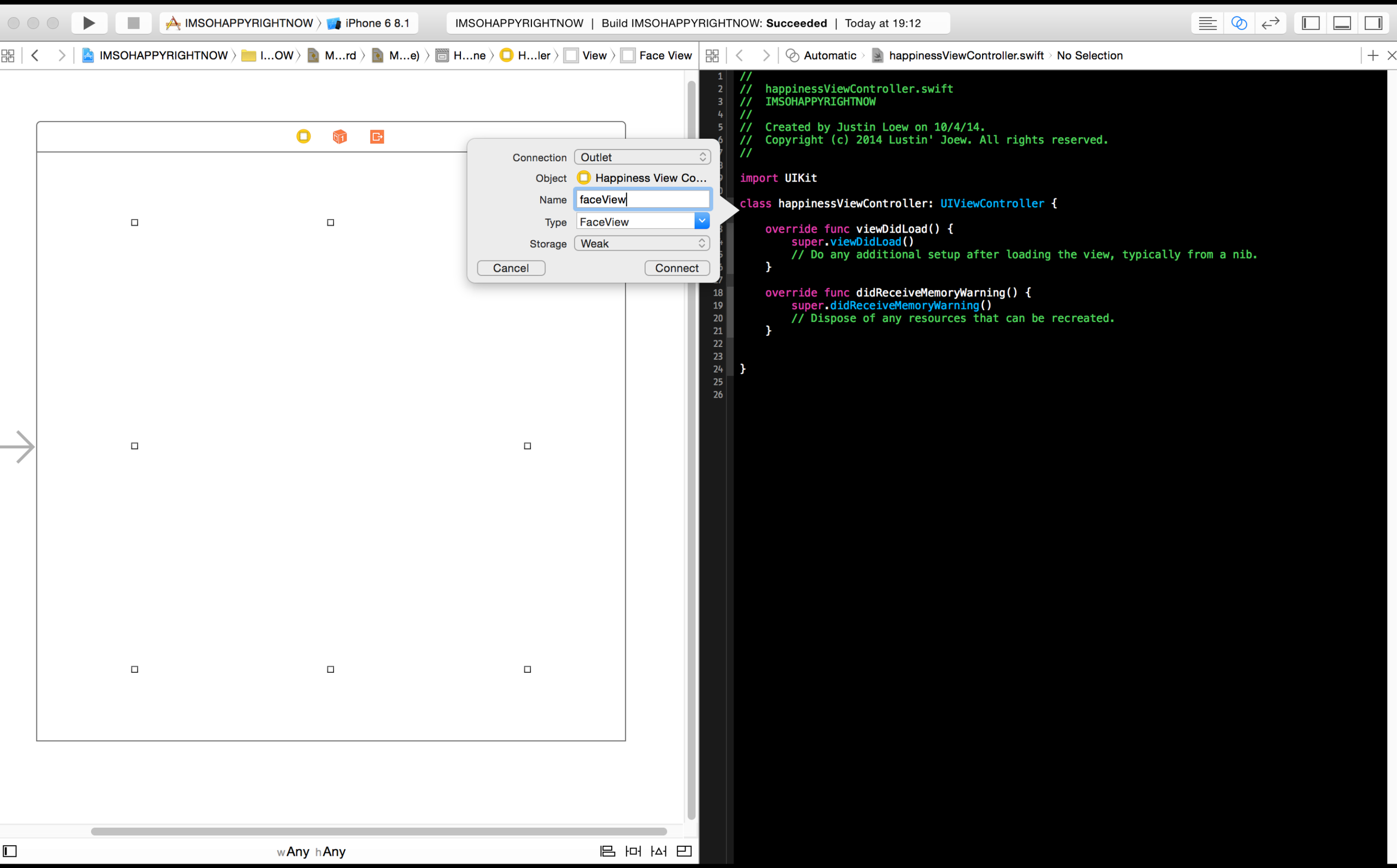
Select the ViewController, and in the Utilities pane on the right, set the class of the ViewController to happinessViewController.



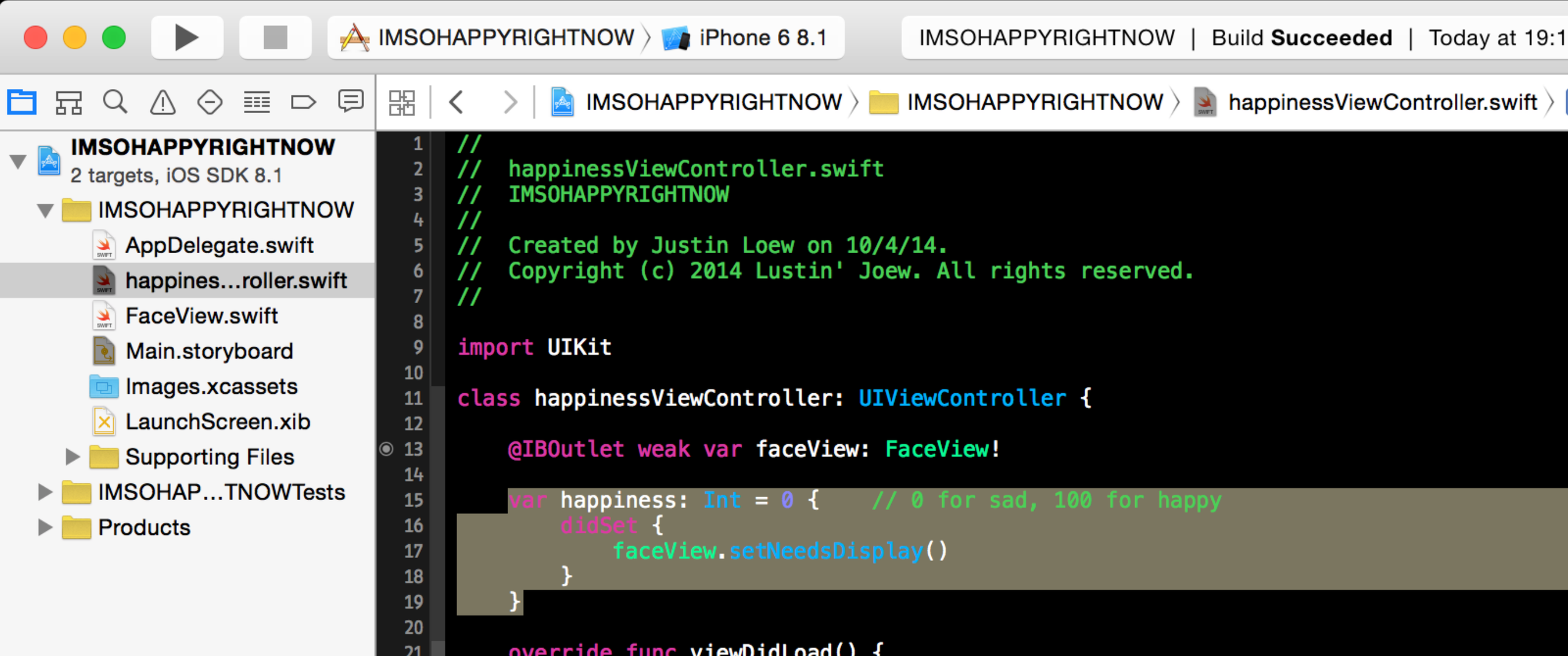
In the bottom right, find a `View` and drag it into the middle of the `happinessViewController`. Set its class to `FaceView`.



Click the linked rings in the top right (where the butler used to be). Right click and drag from the `View` on the left to the code on the right as shown.



Click the button left of the linked rings to switch back to normal mode, then switch to `happinessViewController.swift`.




```swift
//
//  happinessViewController.swift
//  IMSOHAPPYRIGHTNOW
//
//  Created by Justin Loew on 10/4/14.
//  Copyright (c) 2014 Lustin' Joew. All rights reserved.
//

import UIKit

class happinessViewController: UIViewController {

    @IBOutlet weak var faceView: FaceView!

    var happiness: Int = 0 {    // 0 for sad, 100 for happy
        didSet {
            faceView.setNeedsDisplay()
        }
    }

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view, typically from a nib.
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be recreated.
    }


}
```

didSet is automatically called after `faceView` is set.

IMSOHAPPYRIGHTNOW 〉 IMSOHAPPYRIGHTNOW 〉 happinessViewController.swift 〉 happinessViewController

Make sure `happinessViewController` implements our `FaceViewDataSource` protocol.

```
1  //
2  //  happinessViewController.swift
3  //  IMSOHAPPYRIGHTNOW
4  //
5  //  Created by Justin Loew on 10/4/14.
6  //  Copyright (c) 2014 Lustin' Joew. All rights reserved.
7  //
8
9  import UIKit
10
11 class happinessViewController: UIViewController, FaceViewDataSource {
12
13     @IBOutlet weak var faceView: FaceView! {
14         didSet {
15             faceView.dataSource = self
16
17             // enable pinch gestures in the FaceView using its pinch() handler
18             faceView.addGestureRecognizer(UIPinchGestureRecognizer(target: faceView, action: "pinch:"))
19
20             faceView.addGestureRecognizer(UIPanGestureRecognizer(target: self, action: "handleHappinessChange:"))
21         }
22     }
23
24     var happiness: Int = 0 {    // 0 for sad, 100 for happy
25         didSet {
26             faceView.setNeedsDisplay()
27         }
28     }
```

IMSOHAPPYRIGHTNOW
2 targets, iOS SDK 8.1
- IMSOHAPPYRIGHTNOW
  - AppDelegate.swift
  - happines...roller.swift
  - FaceView.swift
  - Main.storyboard
  - Images.xcassets
  - LaunchScreen.xib
  - Supporting Files
- IMSOHAP...TNOWTests
- Products


```swift
//
//  happinessViewController.swift
//  IMSOHAPPYRIGHTNOW
//
//  Created by Justin Loew on 10/4/14.
//  Copyright (c) 2014 Lustin' Joew. All rights reserved.
//

import UIKit

class happinessViewController: UIViewController, FaceViewDataSource {

    @IBOutlet weak var faceView: FaceView! {
        didSet {
            faceView.dataSource = self

            // enable pinch gestures in the FaceView using its pinch() handler
            faceView.addGestureRecognizer(UIPinchGestureRecognizer(target: faceView, action: "pinch"))

            faceView.addGestureRecognizer(UIPanGestureRecognizer(target: self, action: "handleHappinessChange"))
        }
    }

    var happiness: Int = 0 {    // 0 for sad, 100 for happy
        didSet {
            faceView.setNeedsDisplay()
        }
    }

    func handleHappinessChange(gesture: UIPanGestureRecognizer) {
        if gesture.state == .Changed || gesture.state == .Ended {
            let translationAmount: CGPoint = gesture.translationInView(faceView)
            happiness += Int(translationAmount.y / 2)
            if happiness < 0 {
                happiness = 0
            }
            if happiness > 100 {
                happiness = 100
            }

            gesture.setTranslation(CGPointZero, inView: faceView)
        }
    }

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view, typically from a nib.
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be recreated.
    }


}
```

IMSOHAPPYRIGHTNOW
2 targets, iOS SDK 8.1
- IMSOHAPPYRIGHTNOW
  - AppDelegate.swift
  - happines...roller.swift
  - FaceView.swift
  - Main.storyboard
  - Images.xcassets
  - LaunchScreen.xib
  - Supporting Files
- IMSOHAP...TNOWTests
- Products

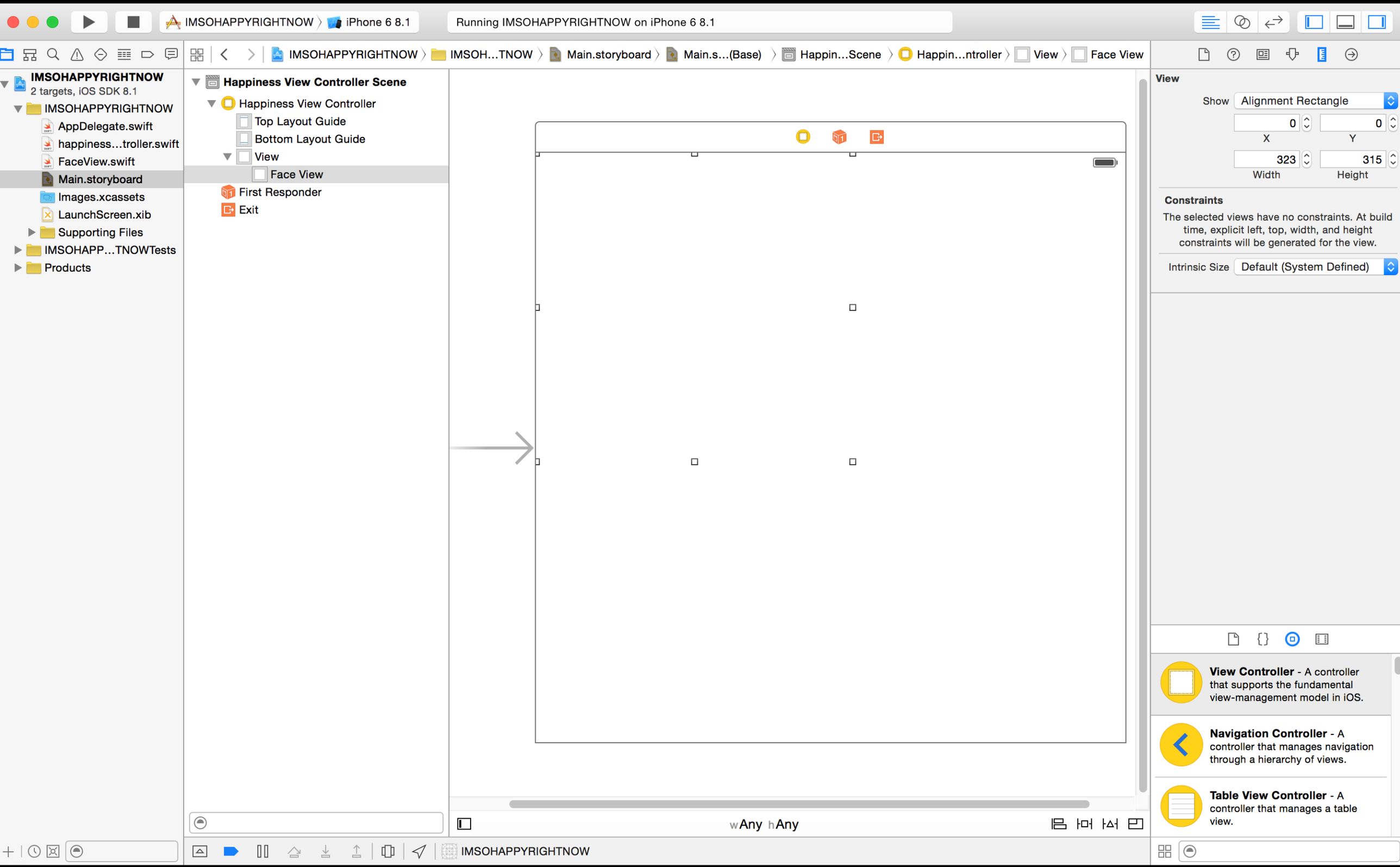

```swift
//
//  happinessViewController.swift
//  IMSOHAPPYRIGHTNOW
//
//  Created by Justin Loew on 10/4/14.
//  Copyright (c) 2014 Lustin' Joew. All rights reserved.
//

import UIKit

class happinessViewController: UIViewController, FaceViewDataSource {

    @IBOutlet weak var faceView: FaceView! {
        didSet {
            faceView.dataSource = self

            // enable pinch gestures in the FaceView using its pinch() handler
            faceView.addGestureRecognizer(UIPinchGestureRecognizer(target: faceView, action: "pinch"))

            faceView.addGestureRecognizer(UIPanGestureRecognizer(target: self, action: "handleHappinessChange"))
        }
    }

    var happiness: Int = 0 {    // 0 for sad, 100 for happy
        didSet {
            faceView.setNeedsDisplay()
        }
    }

    func handleHappinessChange(gesture: UIPanGestureRecognizer) {
        if gesture.state == .Changed || gesture.state == .Ended {
            let translationAmount: CGPoint = gesture.translationInView(faceView)
            happiness += Int(translationAmount.y / 2)
            if happiness < 0 {
                happiness = 0
            }
            if happiness > 100 {
                happiness = 100
            }

            gesture.setTranslation(CGPointZero, inView: faceView)
        }
    }

    func smileForFaceView(sender: FaceView) -> Float {
        // happiness is 0-100. smile range is -1 to 1.
        return Float(happiness - 50) / Float(50)
    }

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view, typically from a nib.
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be recreated.
    }


}
```

IMSOHAPPYRIGHTNOW
2 targets, iOS SDK 8.1
- IMSOHAPPYRIGHTNOW
  - AppDelegate.swift
  - happines...roller.swift
  - FaceView.swift
  - Main.storyboard
  - Images.xcassets
  - LaunchScreen.xib
  - Supporting Files
- IMSOHAP...TNOWTests
- Products

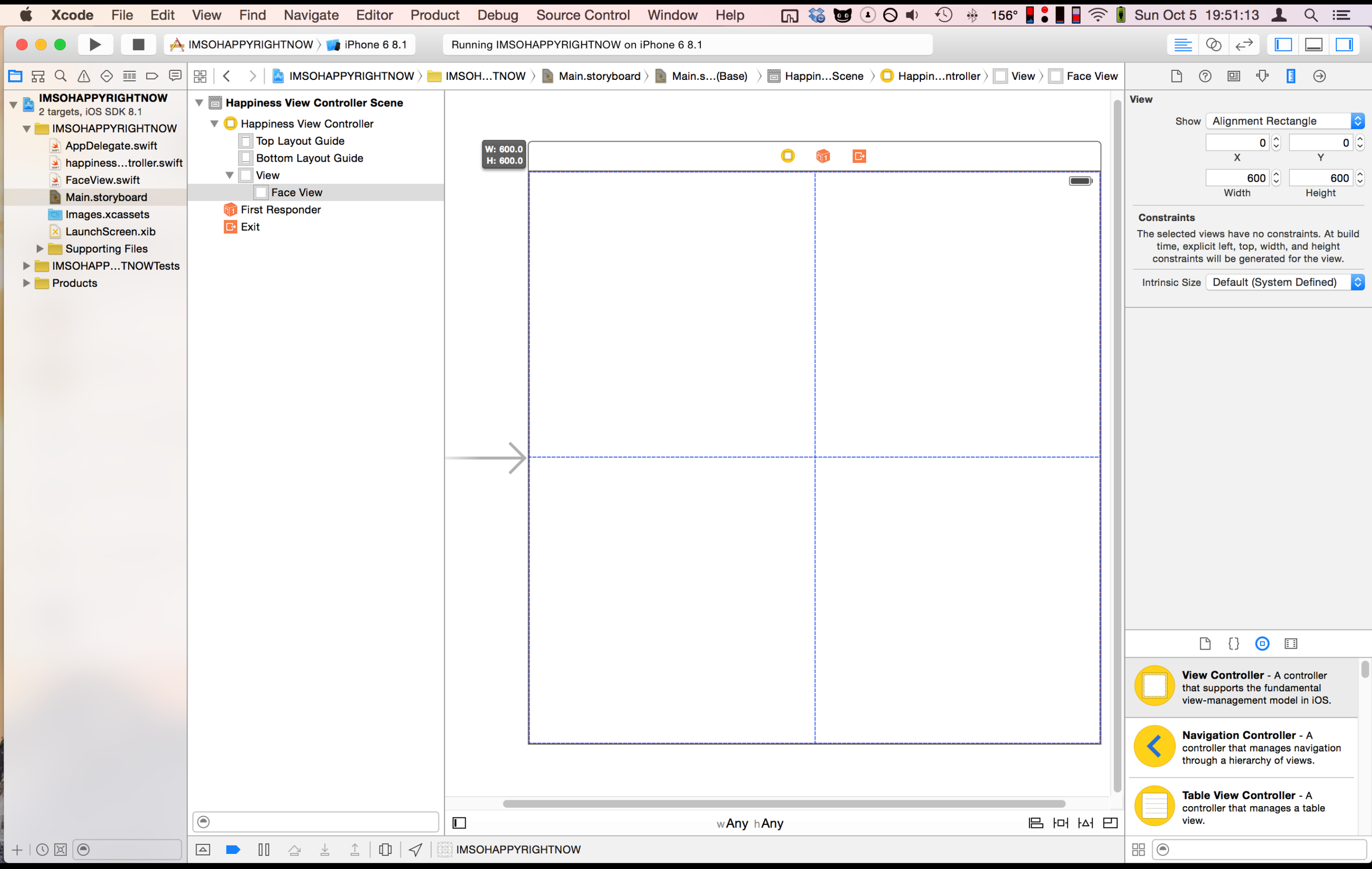

```swift
//
//  happinessViewController.swift
//  IMSOHAPPYRIGHTNOW
//
//  Created by Justin Loew on 10/4/14.
//  Copyright (c) 2014 Lustin' Joew. All rights reserved.
//

import UIKit

class happinessViewController: UIViewController, FaceViewDataSource {

    @IBOutlet weak var faceView: FaceView! {
        didSet {
            faceView.dataSource = self

            // enable pinch gestures in the FaceView using its pinch() handler
            faceView.addGestureRecognizer(UIPinchGestureRecognizer(target: faceView, action: "pinch"))

            faceView.addGestureRecognizer(UIPanGestureRecognizer(target: self, action: "handleHappinessChange"))
        }
    }

    var happiness: Int = 0 {    // 0 for sad, 100 for happy
        didSet {
            faceView.setNeedsDisplay()
        }
    }

    func handleHappinessChange(gesture: UIPanGestureRecognizer) {
        if gesture.state == .Changed || gesture.state == .Ended {
            let translationAmount: CGPoint = gesture.translationInView(faceView)
            happiness += Int(translationAmount.y / 2)
            if happiness < 0 {
                happiness = 0
            }
            if happiness > 100 {
                happiness = 100
            }

            gesture.setTranslation(CGPointZero, inView: faceView)
        }
    }

    func smileForFaceView(sender: FaceView) -> Float {
        // happiness is 0-100. smile range is -1 to 1.
        return Float(happiness - 50) / Float(50)
    }

    override func shouldAutorotate() -> Bool {
        return true // support all orientations
    }

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view, typically from a nib.
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be recreated.
    }
```

`shouldAutorotate` is called to see if the app should turn to landscape mode when the device is tilted.

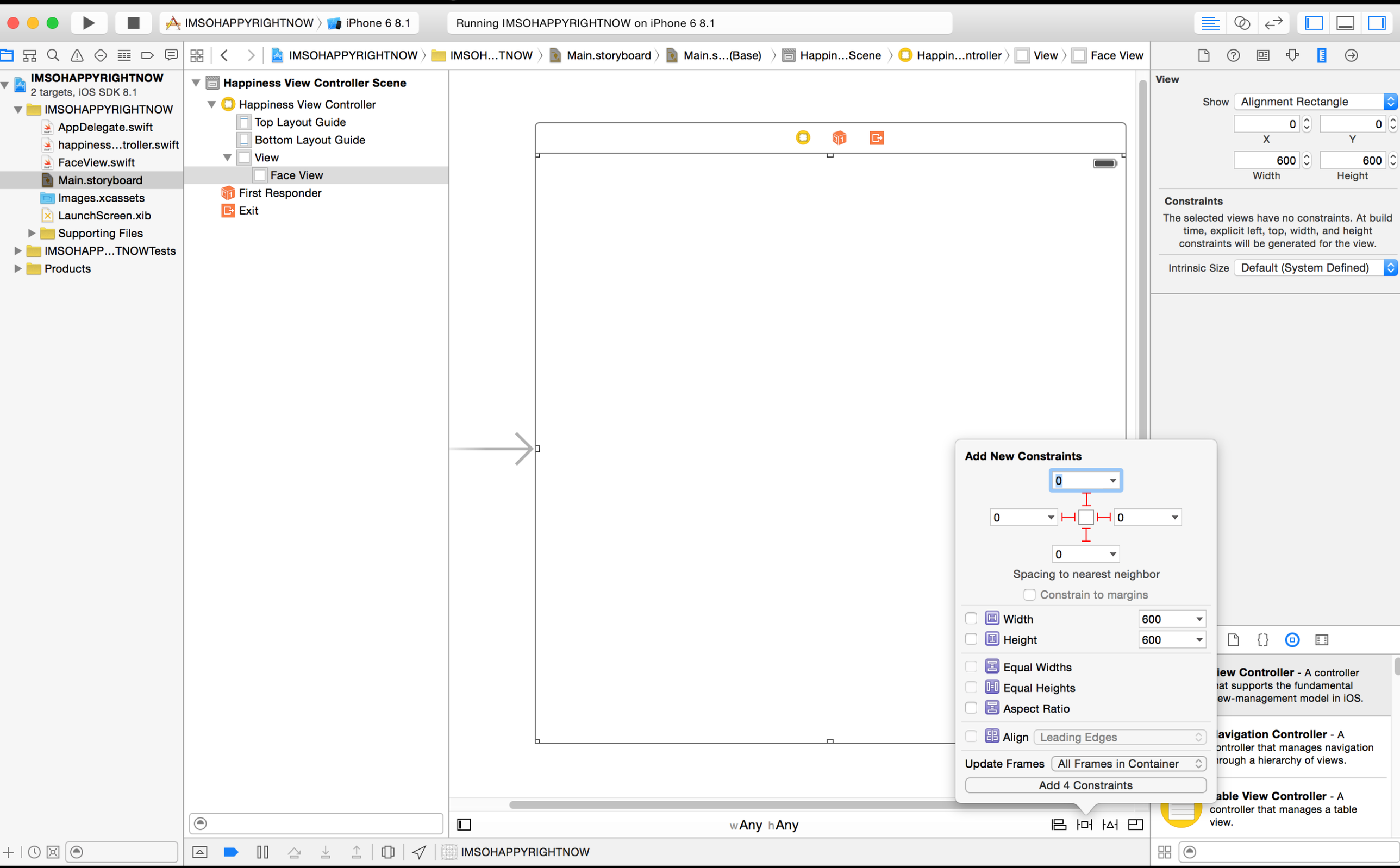
Move the view to the top left corner



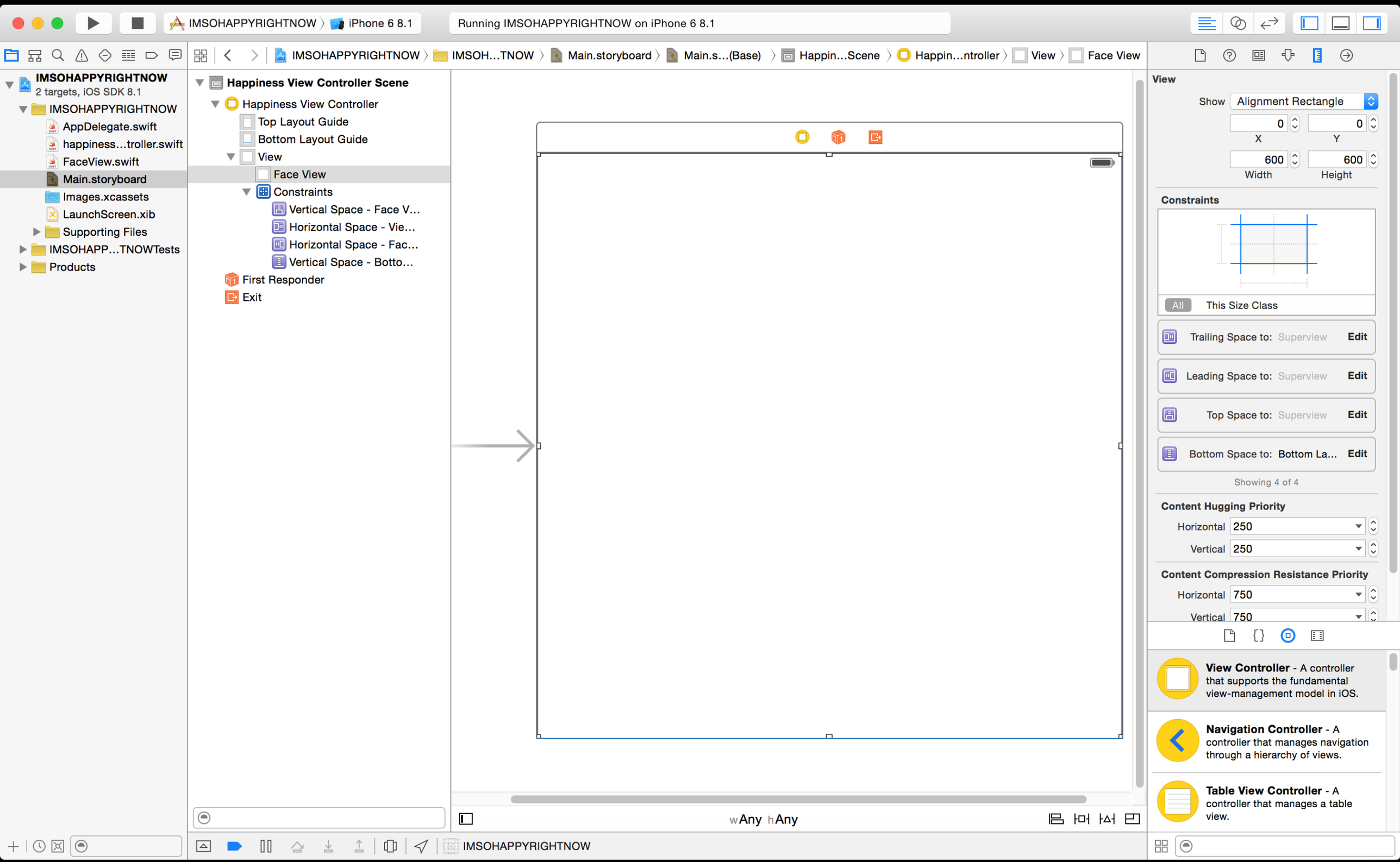
…And expand it to fill the screen.



Use this button to add constraints, so the smiley can autoresize itself for different device sizes. Make sure you click the I-shaped things at the top so they're solid, not dashed.



They should show up in the bar on the left if you did it right.



Great success! Build and Run, and play around with it. Swipe up and down to change the smile to a frown, and pinch to zoom (hold option to get two touches in the simulator).