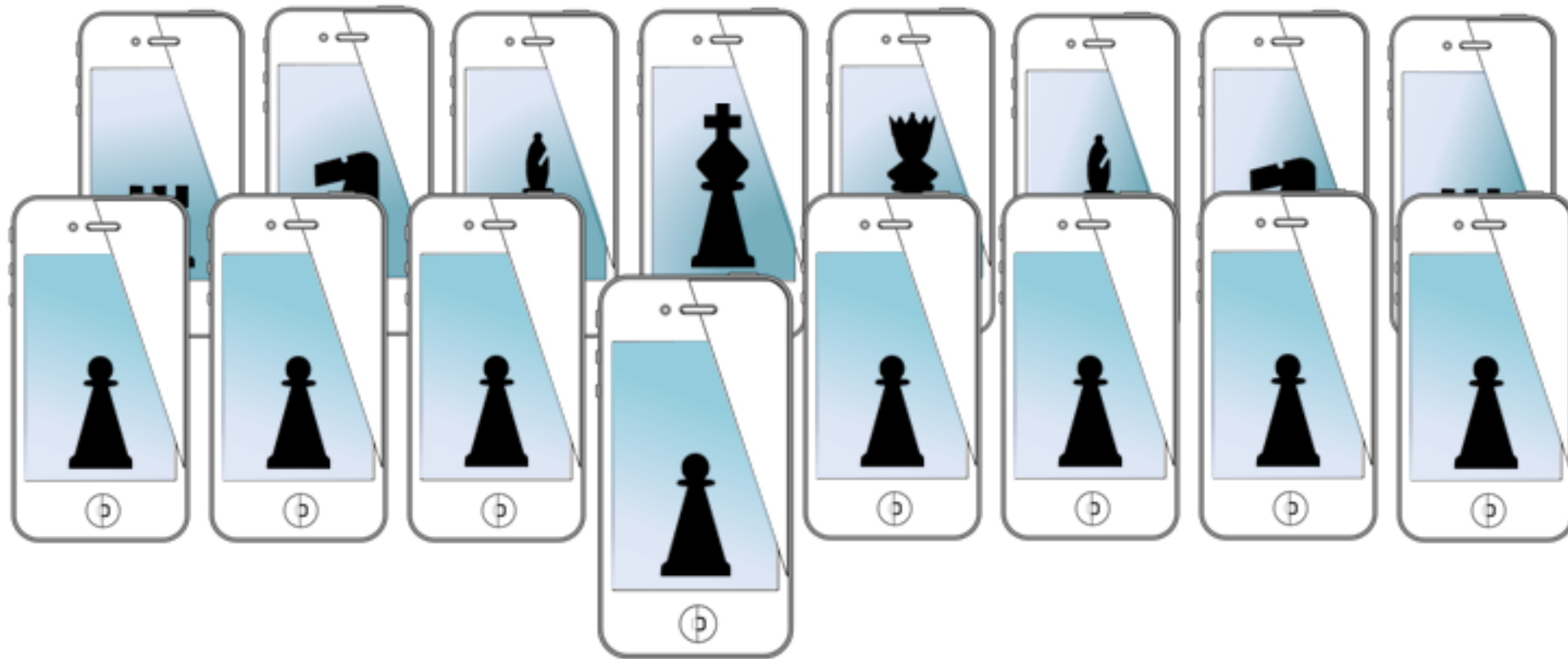


# MOBILE SENSING & LEARNING



## CSE5323 & 7323

Mobile Sensing & Learning

Video Module One, model view controllers

Eric C. Larson, Lyle School of Engineering,  
Computer Science and Engineering, Southern Methodist University

# agenda (video)

- MVC review
  - outlets, actions, delegates, protocols, data source
- ViewControllers in iOS
  - TableViewController, NavigationViewController, CollectionViewController, UIViewController
- storyboard (with UIViewController)
  - outlets, auto layout, programatic creation
  - timers, UIScrollView, image assets,

# MVC's

review

controller has direct connection to view class

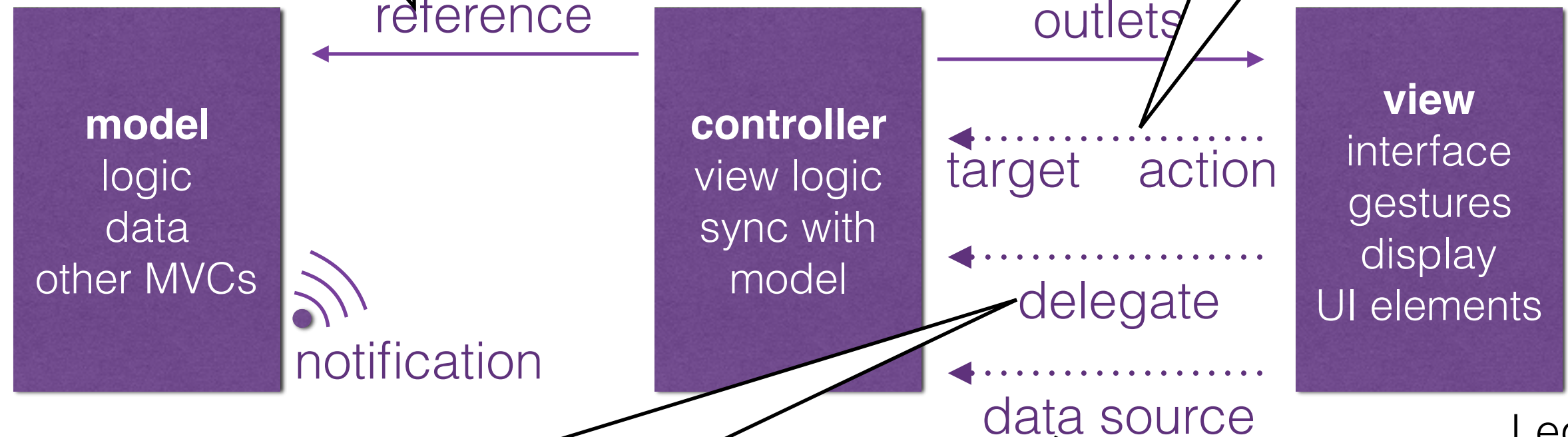
```
@property (weak, nonatomic) IBOutlet UITextField *firstName;
@property (weak, nonatomic) IBOutlet UITextField *lastName;
@property (weak, nonatomic) IBOutlet UITextField *phoneNumber;
```

controller has direct connection to model class

```
ModelClass *myModel = [get global handle to model]
PhoneNumberStruct * phNumber = [myModel getNumber];
self.phoneNumberLabel.text = phNumber.number;
```

view sends a targeted message

```
- (IBAction)buttonPressed:(id)sender;
- (IBAction)showPhBookPressed:(id)sender;
```

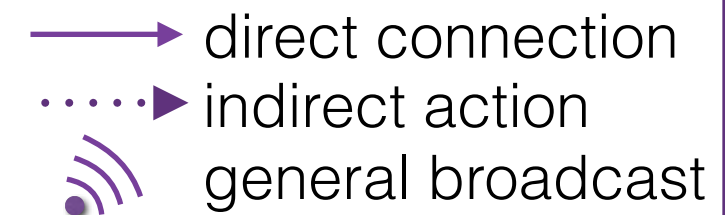


```
MainViewController ()<UITextFieldDelegate>
#pragma mark - UITextField Delegate
- (BOOL)textFieldShouldReturn:(UITextField *)textField { ... }
```

controller implements method for view class

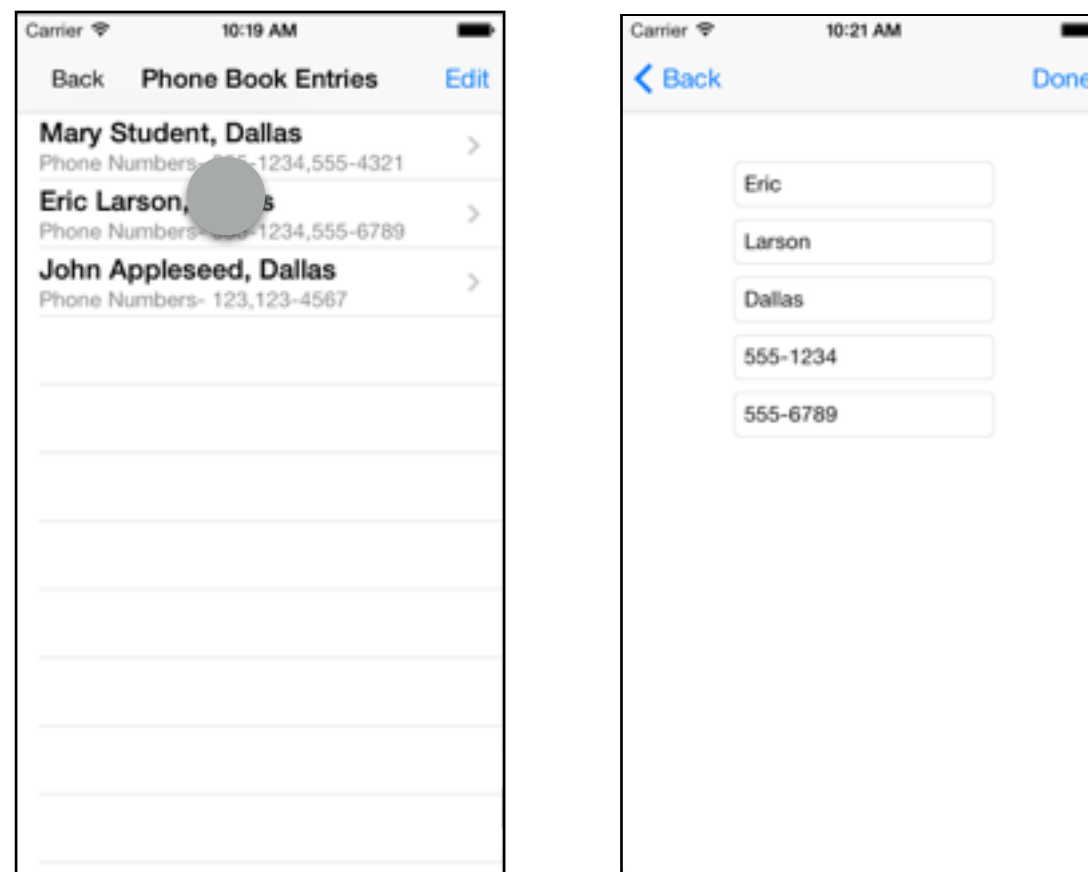
```
- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView
- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSectionSection:(NSInteger)section
```

Legend



# controller life cycle review

- problem: we need to handoff control of the screen to a new view
- the app itself is handling most of this transition
  - app will “unfreeze” the new view and its class properties
- **you** need to send information from **source** ViewController to **destination** ViewController



# controller life cycle review

Source Controller

Destination Controller

view is unfrozen, property memory allocated

`prepareForSegue`  
prepare to leave the screen  
set properties of destination, if needed

view outlets are ready for interaction

`viewDidLoad`

`viewWillAppear`

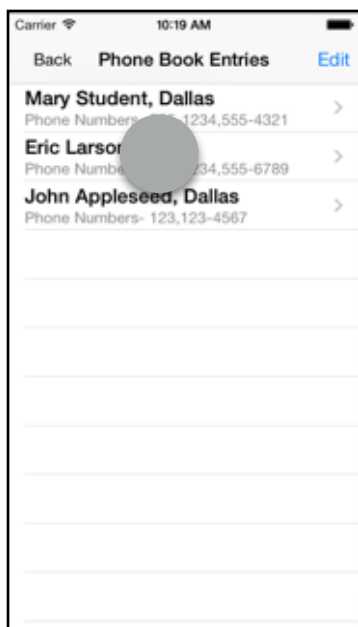
`viewDidAppear`

`viewWillDisappear`

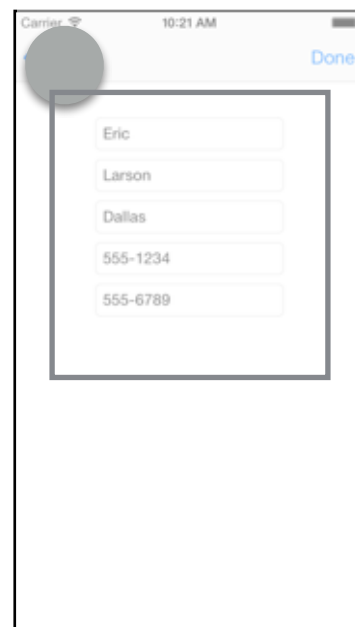
`viewDidDisappear`

memory deallocated when app is ready

source



destination

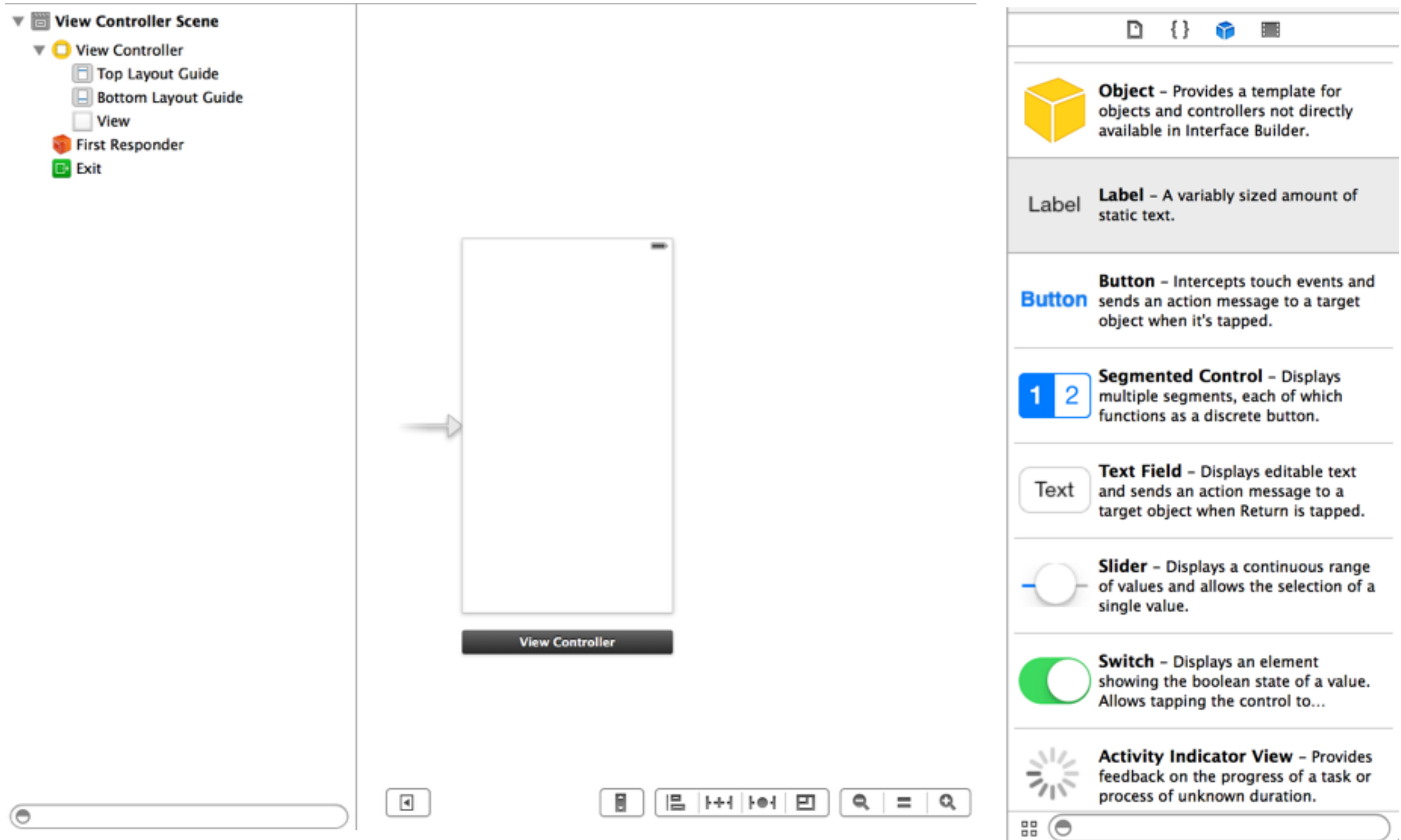


**user**

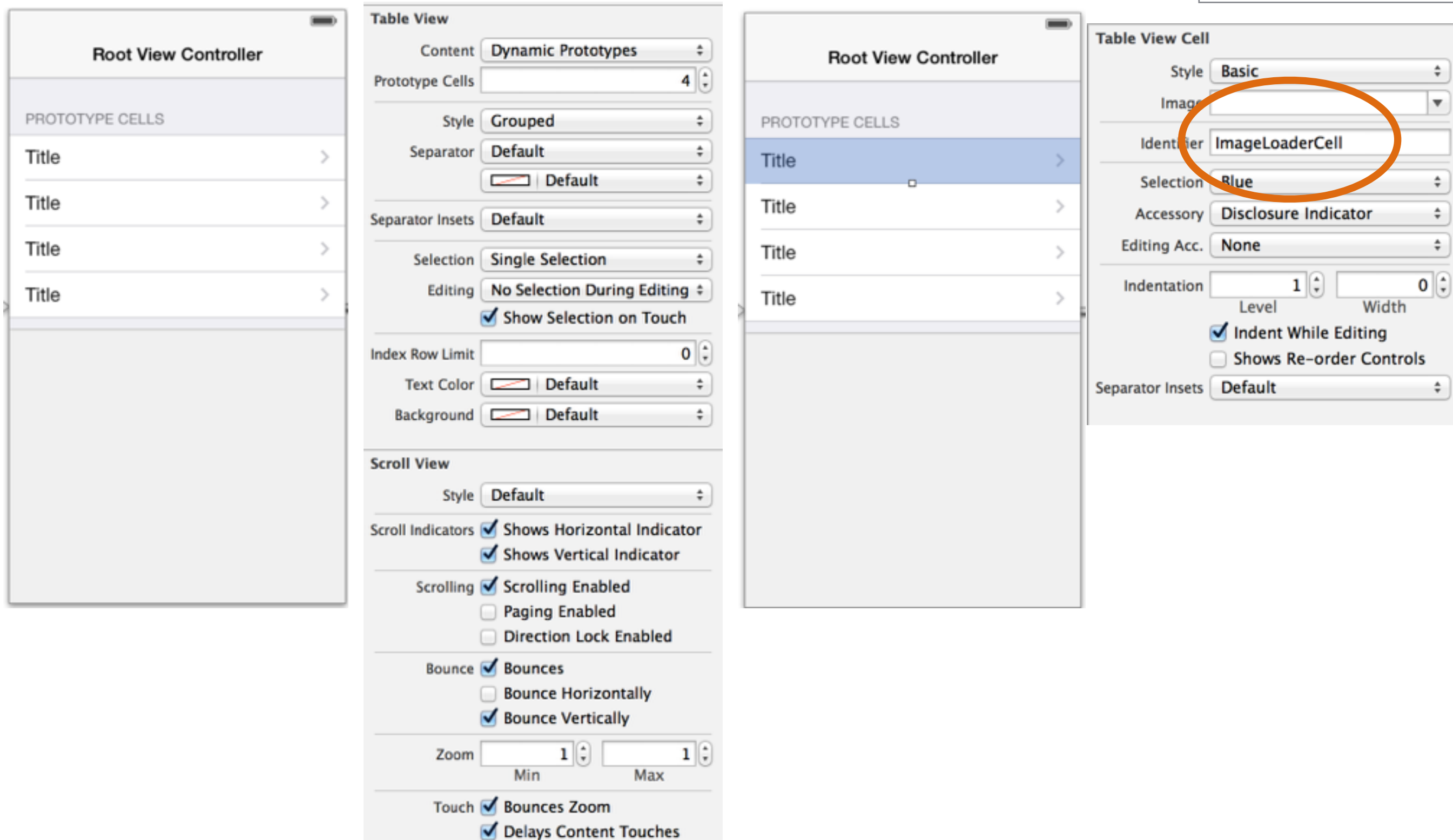




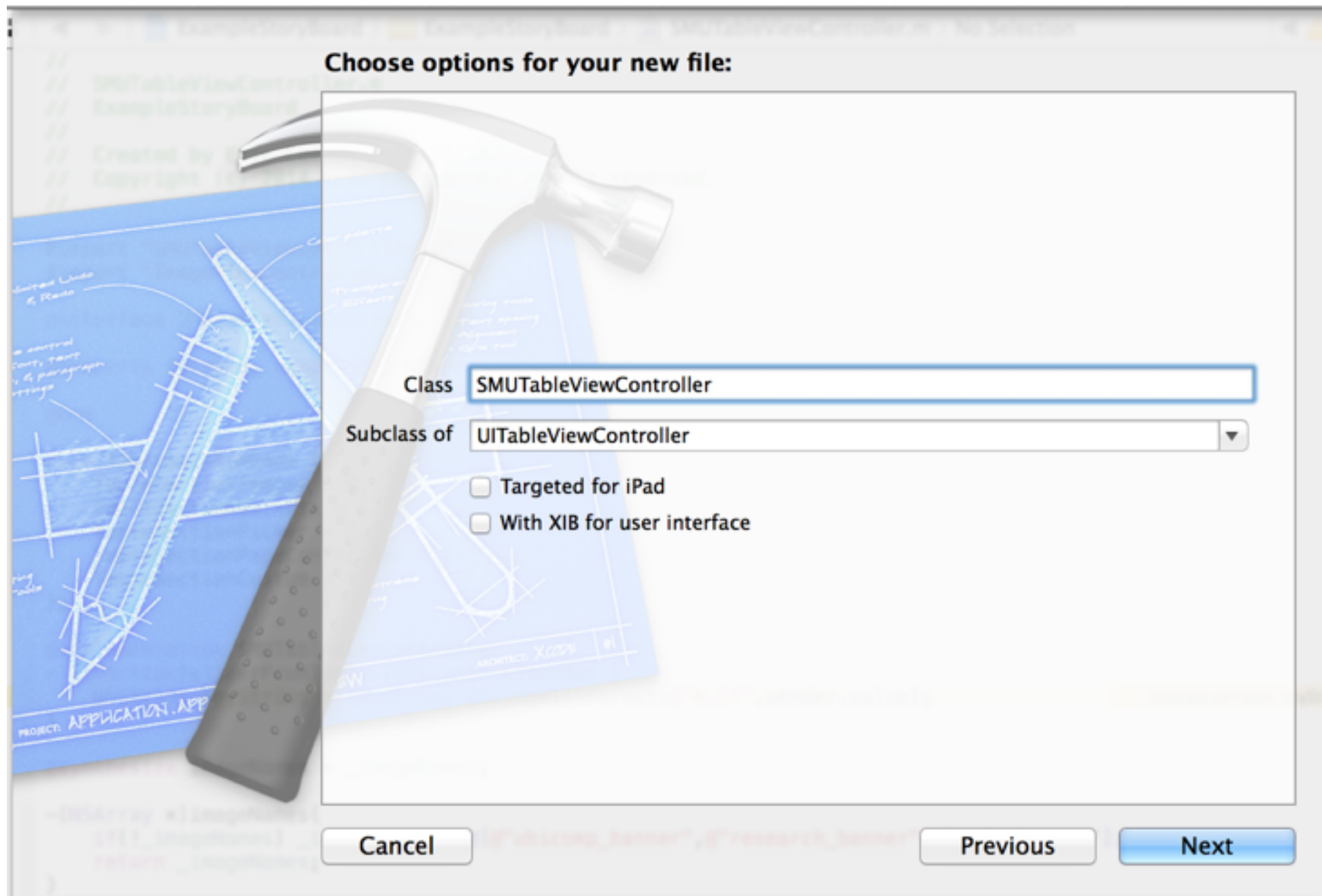
# the storyboard



# table view controller



# table view controller





# table view controller

- must implement “data source” methods

```
- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView
{
    return numSections;
}

- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSectionSection:(NSInteger)section
{
    return rowsInSectionNumber[section];
}

- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath
{
    static NSString *CellIdentifier = nil;
    UITableViewCell *cell = nil;

    CellIdentifier = @"ImageLoaderCell";
    cell = [tableView dequeueReusableCellWithIdentifier:CellIdentifier forIndexPath:indexPath];

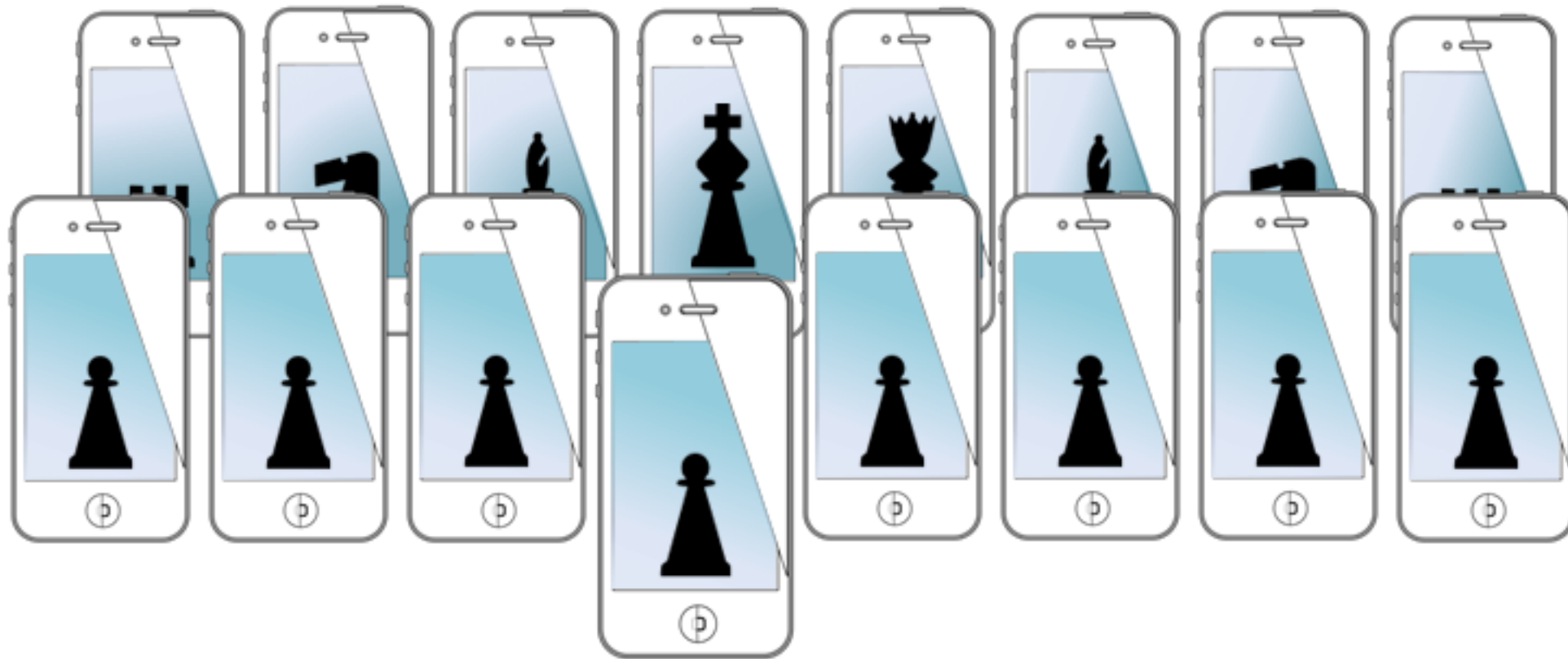
    // Configure the cell
    cell.textLabel.text = @"An Image";
    return cell;
}
```

cell prototype from storyboard

set cell attributes

# table view controller demo

# MOBILE SENSING & LEARNING



## CSE5323 & 7323

Mobile Sensing & Learning

Video Module One, model view controllers

Eric C. Larson, Lyle School of Engineering,  
Computer Science and Engineering, Southern Methodist University

# scroll view delegate

```
@interface SomeViewController () <UIScrollViewDelegate>
```

add view to the scroll view

```
[self.someScrollView addSubview:self.imageView];  
self.someScrollView.contentSize = self.imageView.size;  
self.someScrollView.minimumZoomScale = 0.1;  
self.someScrollView.delegate = self;
```

I am a delegate for the Scroll View: I implement methods in the Scroll View Protocol!

set VC as delegate

```
#pragma Delegate Methods  
-(UIView*) viewForZoomingInScrollView:(UIScrollView *)scrollView  
{  
    return self.imageView;  
}
```

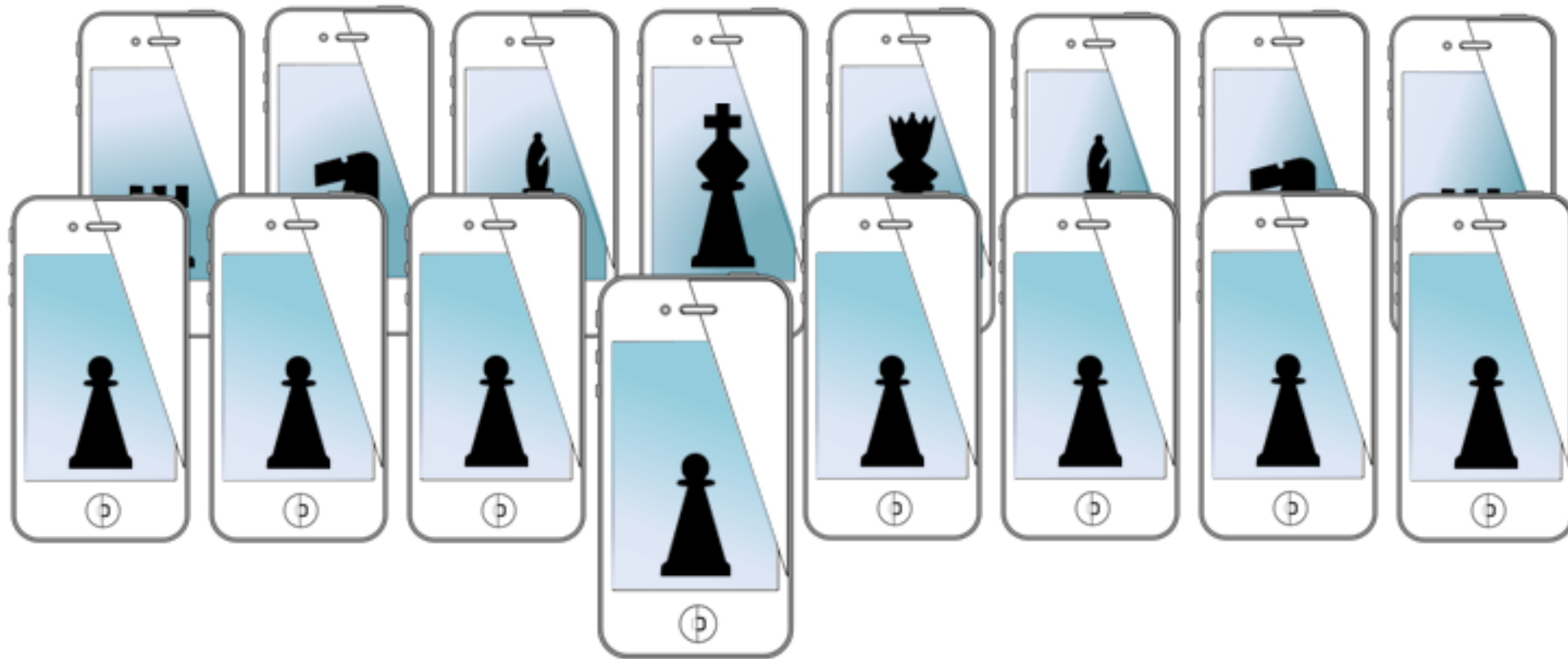
one of many methods in the protocol

# demo





# MOBILE SENSING & LEARNING



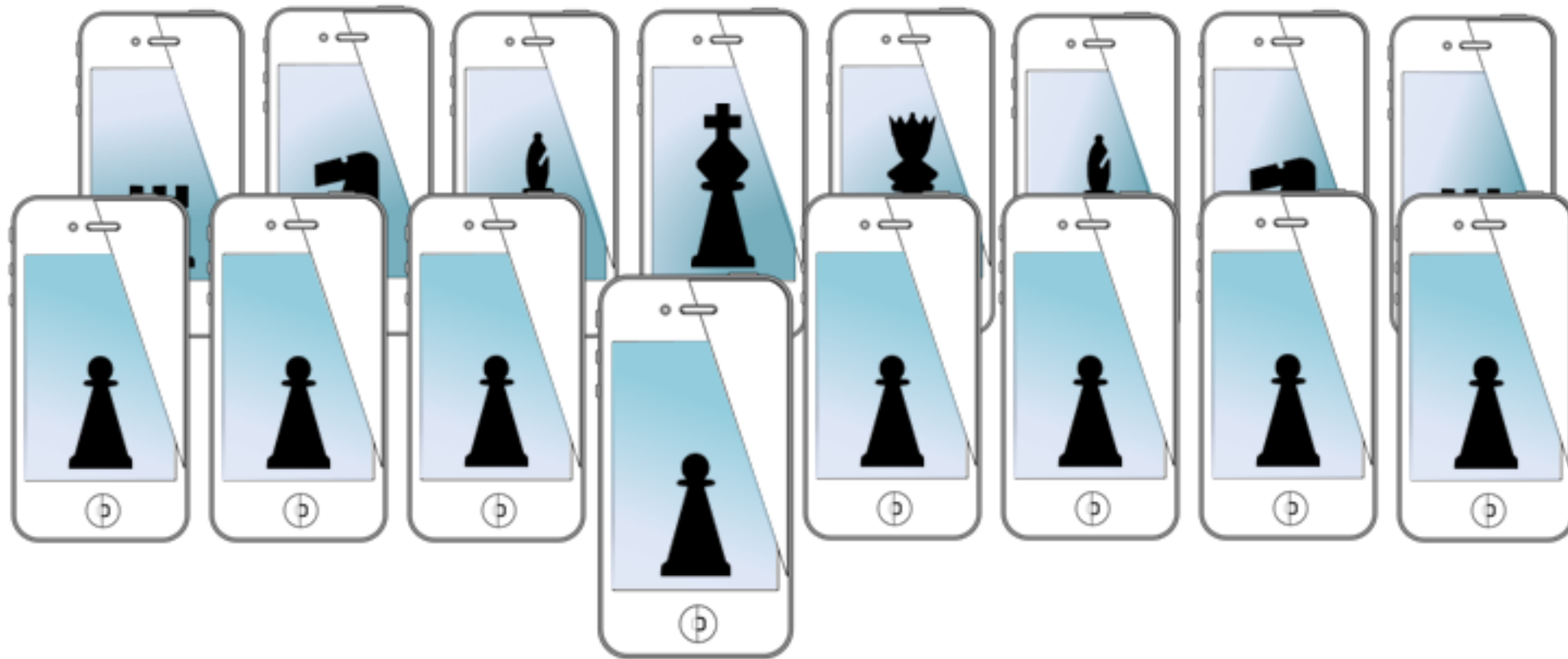
## CSE5323 & 7323

Mobile Sensing & Learning

Video Module One, model view controllers

Eric C. Larson, Lyle School of Engineering,  
Computer Science and Engineering, Southern Methodist University

# MOBILE SENSING & LEARNING



## CSE5323 & 7323

Mobile Sensing & Learning

week two, lecture three: UI elements swift

Eric C. Larson, Lyle School of Engineering,  
Computer Science and Engineering, Southern Methodist University

# course logistics

- University developer program is a go!
  - send me an email if you want an invite
- A1 due next Friday
  - come show me the app in my office
  - questions?

# agenda

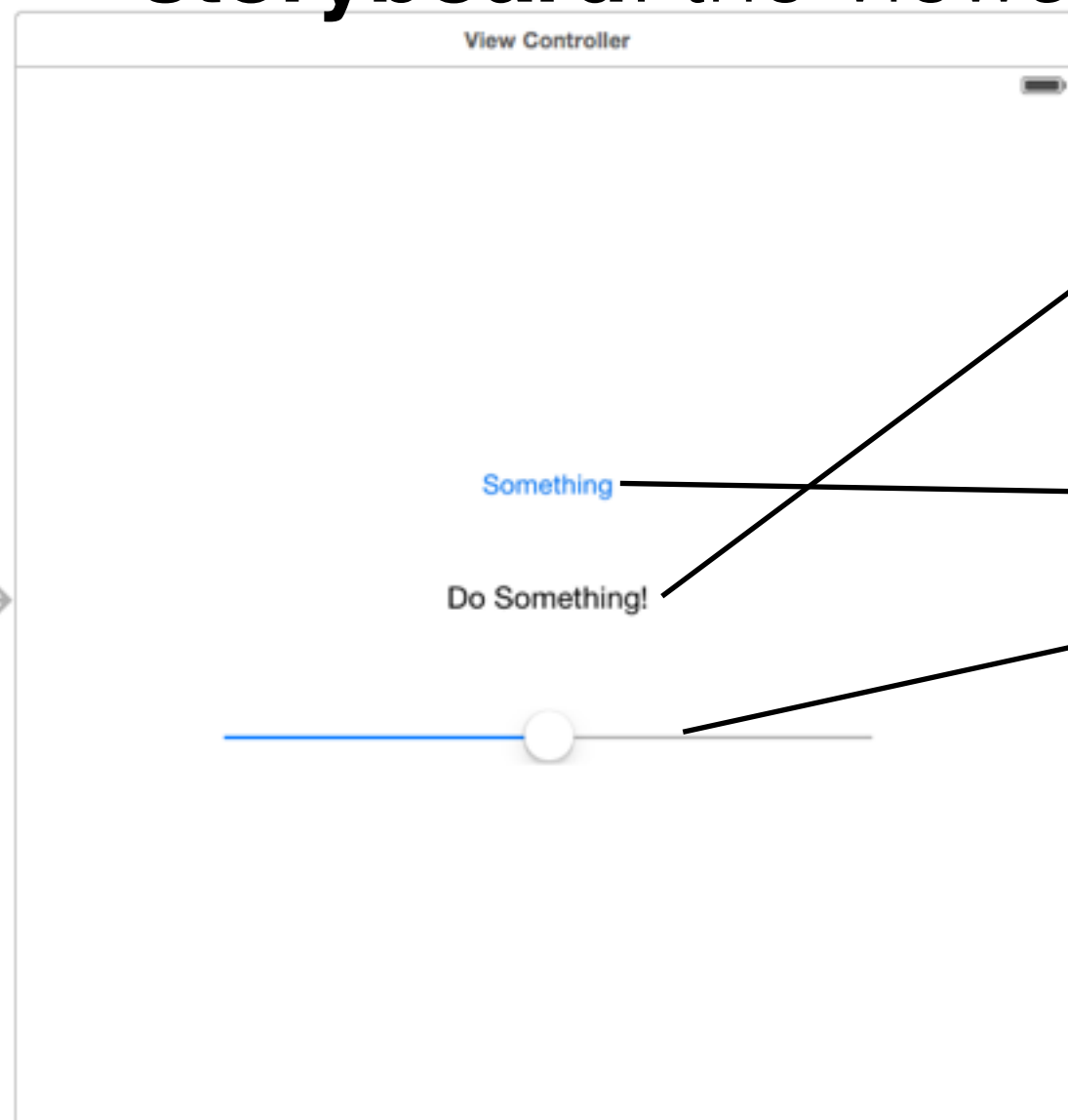
- target action behavior
  - and constraints
- text fields
- gesture recognizers
- timers / segmented control
- **if time:** swift!

# target and action

- UI elements that need to communicate back with their controllers will generate actions

**storyboard:** the views

**class:** the controller

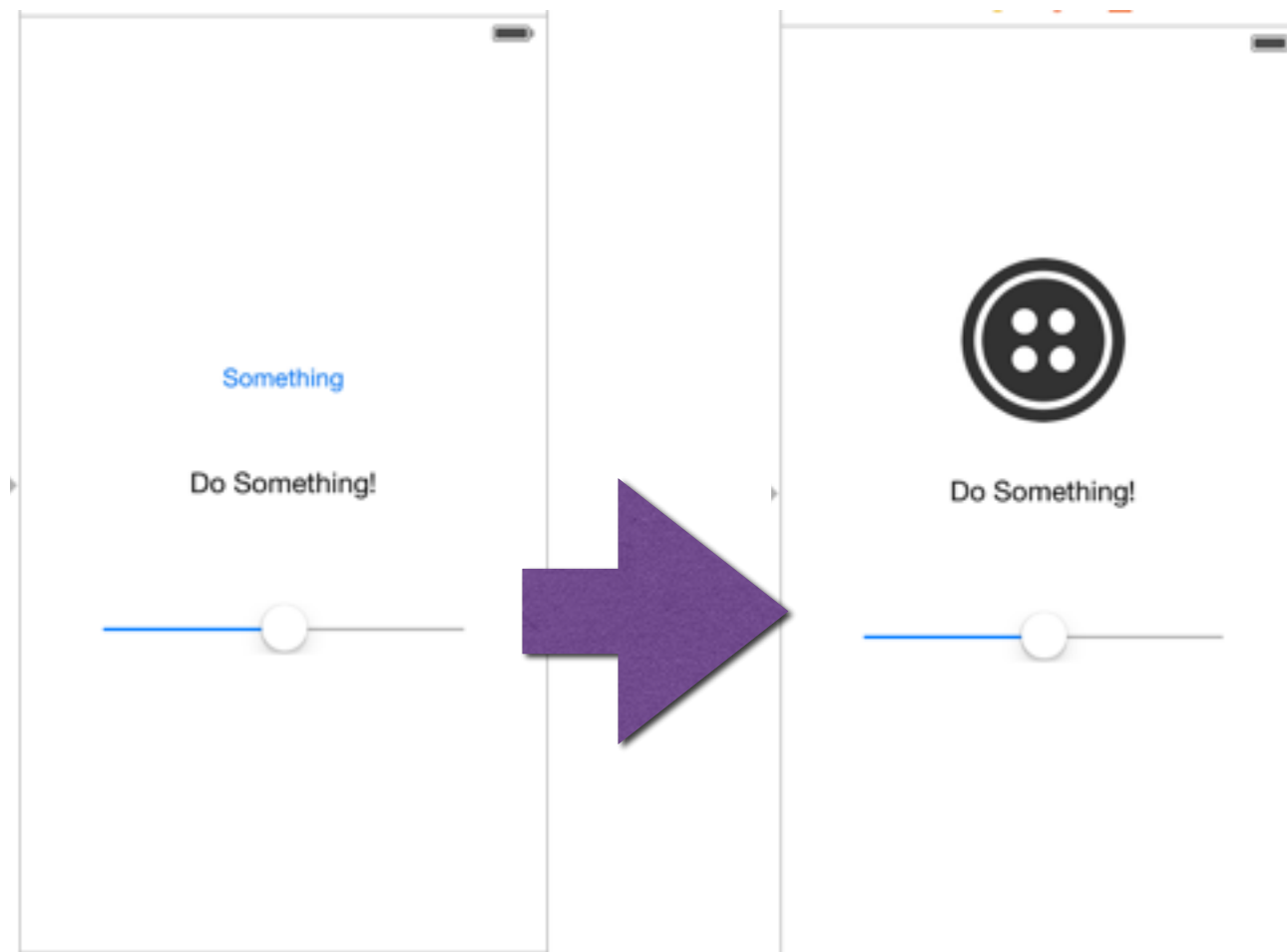


```
8 //
9 #import "ViewController.h"
10
11 @interface ViewController ()
12 @property (weak, nonatomic) IBOutlet UILabel *
13     somethingLabel;
14 @end
15
16 @implementation ViewController
17 - (IBAction)buttonPressed:(UIButton *)sender {
18     self.somethingLabel.text = @"Thanks!";
19 }
20 - (IBAction)sliderChanged:(UISlider *)sender {
21     self.somethingLabel.text = [NSString
22         stringWithFormat:@"Value of slider is %.
23         2f",sender.value];
24 }
25
26 - (void)viewDidLoad {
27     [super viewDidLoad];
28     // Do any additional setup after loading the
29     // view, typically from a nib.
30 }
```



# bring your buttons to life

- in many settings you are **given criteria** from a graphic designer
  - but right now, **you** are the graphic designer
- use **images** for more **descriptive** buttons and labels
- **good tip**: make them the right size from the start!

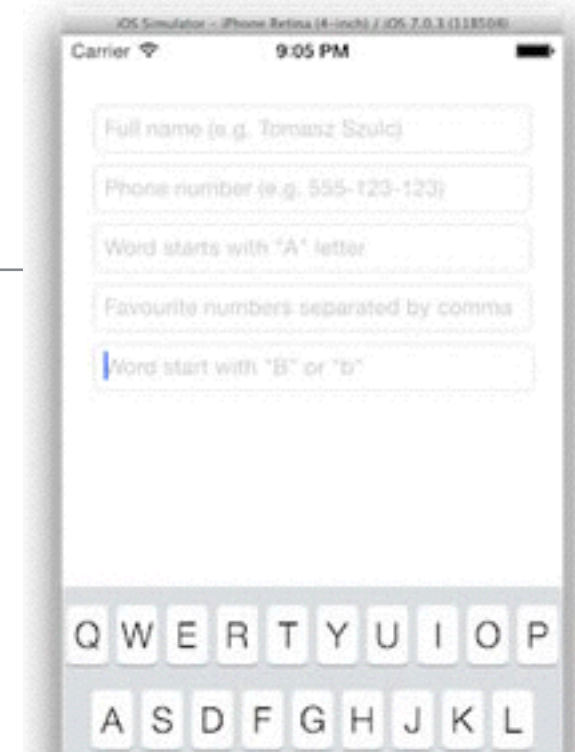


# UI basics demo



# text fields

- text fields are very common input
- but they require the use of the keyboard!
- so you need to become a delegate
  - say when to dismiss the keyboard
  - define what happens to text that the user entered



**outlet**, setup from storyboard

```
@interface ViewController () <UITextFieldDelegate>
@property (weak, nonatomic) IBOutlet UITextField *nameTextField;
@end
```

```
@implementation ViewController
```

return button pressed

```
viewDidLoad {
    [self.view viewDidLoad];
    self.nameTextField.delegate = self;
}

-(BOOL)textFieldShouldReturn:(UITextField *)textField{
    [textField resignFirstResponder];
    return YES;
}
```

tell compiler we are delegate

make VC delegate

give up keyboard control

# UI text field demo



# gesture recognition

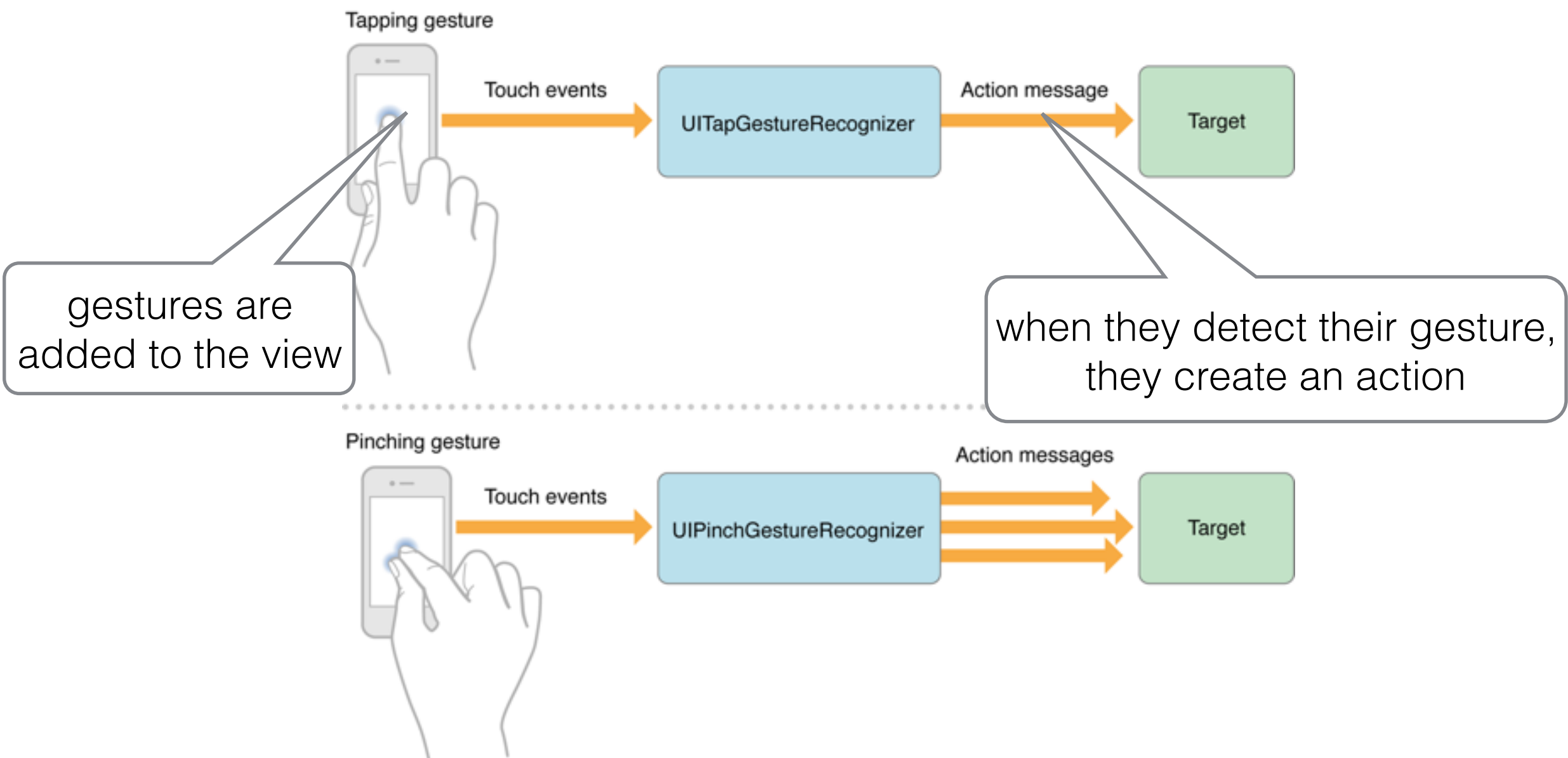
- the fun part about doing things on the iPhone!
- **the point:** recognize different gestures and then make something happen
- lots of ways to do this
  - **programmatically:** quick and versatile
  - **target-action:** easy
  - **delegation:** more feature rich
- here is the complete documentation:

[https://developer.apple.com/library/ios/documentation/EventHandling/Conceptual/EventHandlingiPhoneOS/GestureRecognizer\\_basics/GestureRecognizer\\_basics.html](https://developer.apple.com/library/ios/documentation/EventHandling/Conceptual/EventHandlingiPhoneOS/GestureRecognizer_basics/GestureRecognizer_basics.html)



# gesture recognition

- need a UIGestureRecognizer
- UITapGestureRecognizer, UIPinchGestureRecognizer, ...



# UI gesture demo

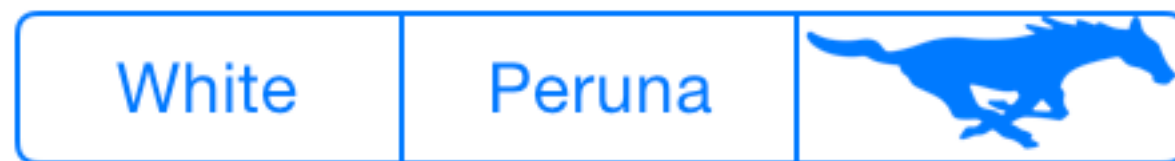


# timers, segmented control

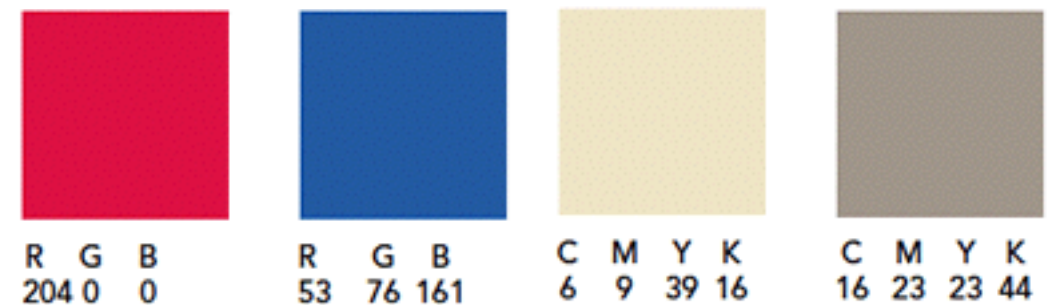
```
- (IBAction)updateFromSegmentedControl:(UISegmentedControl *)sender {  
    NSString *selectedText = [sender titleForSegmentAtIndex: [sender selectedSegmentIndex]];  
    YOUR_CODE  
}
```

get title from control

get value of control



standard SMU colors

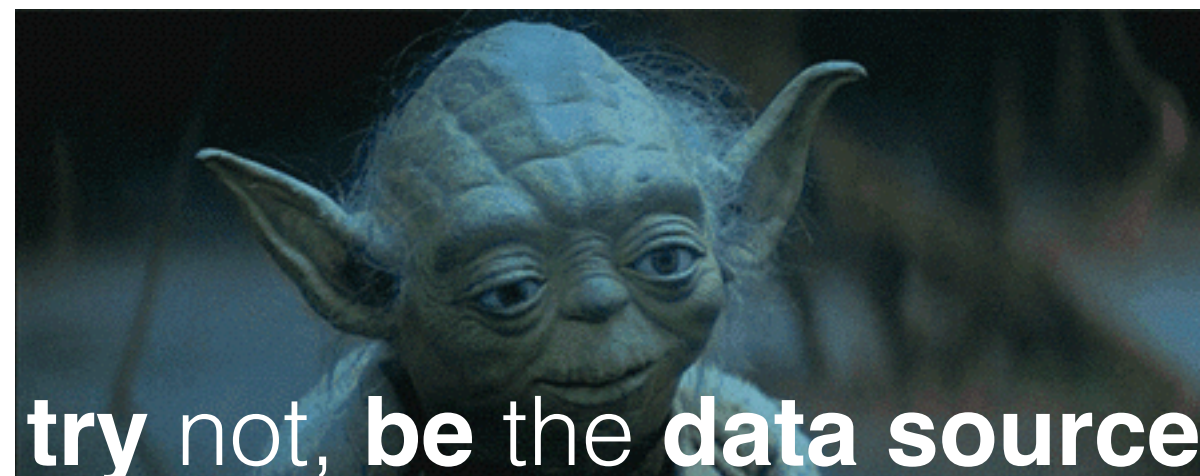


```
NSTimer *timer = [NSTimer scheduledTimerWithTimeInterval:someIntervalInSeconds  
                  target:self  
                  selector:@selector(someFunction:)  
                  userInfo:nil  
                  repeats:YES];  
  
// don't get blocked by the main thread  
[[NSRunLoop mainRunLoop] addTimer:timer forMode:NSRunLoopCommonModes];
```

avoid stops in timer from UI

# pickers

- **look at documentation:** find out how to use a picker view
- you have all the tools to do it from working with collections and the table view controllers!
- you are the data source



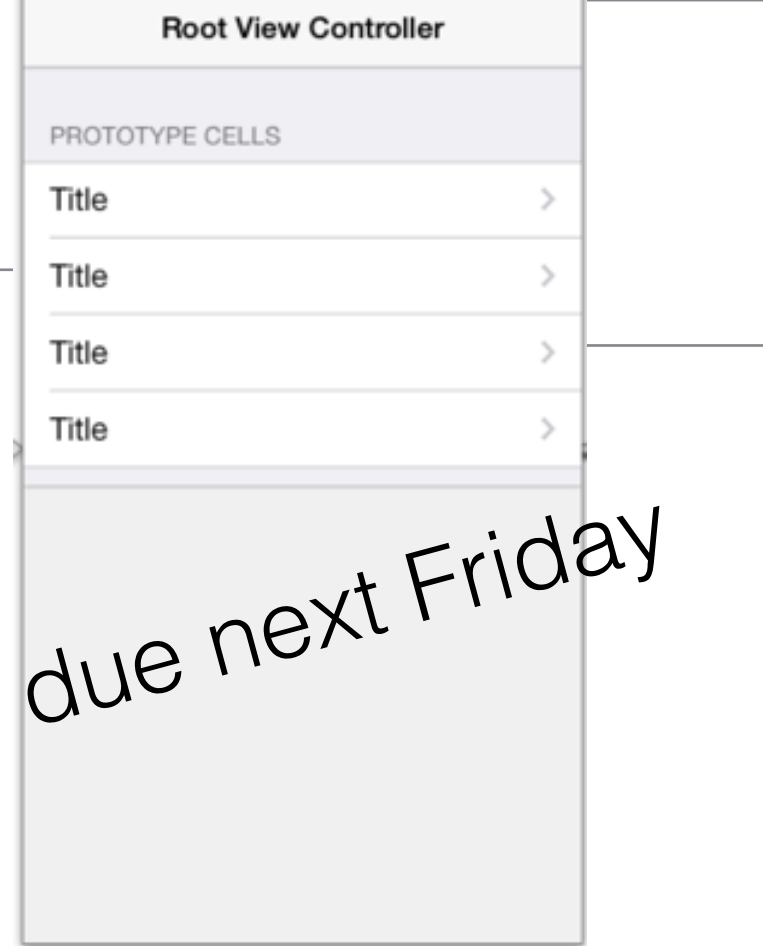
# assignment one

- You have free reign to create an application that manages some type of mutable information: you might display images from online somewhere, stock exchange information, information from twitter--or movies, or books, or amazon
- The data you load and display can come from anywhere and you can do whatever you want with it.
- must use the interface elements as described (**next slide**). You will need to get creative in order to incorporate ALL the design elements below.
- Create an iOS application in XCode that:
  - uses a **TableViewController** to load different views
  - must implement **three different types of cells and load them dynamically** (i.e., you cannot use a static table).
  - View navigation can be hierarchical in any way you want
  - When loading a new view controller your main view controller should hand off information to the controller that is getting created



# assignment one

- Automatic Layout
- Buttons, Sliders, and Labels
- Stepper and Switch
- Picker (you must implement picker delegate)
- Segmented Control
- Timer (which should repeat and somehow update the UIView)
- ScrollView (with scrollable, zoomable content)
- Image View
- Navigation Controller
- Collection View Controller
- Table View Controller
- (Exceptional) Implement a modal view and handle properly using delegation



# for next week...

- concurrency though blocks
- audio sessions
- introduction to sampled data
- graphing audio samples

# if time: slides!

## swift

- syntax is nothing like objective c
- a lot like python syntax (but not)
- weakly typed, no need for semicolons
- can be hard to read or interpret
- powerful use of *tuples, optionals, switch*
- you need to look online for more material than this lecture
  - [https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift\\_Programming\\_Language/GuidedTour.html#//apple\\_ref/doc/uid/TP40014097-CH2-ID1](https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/GuidedTour.html#//apple_ref/doc/uid/TP40014097-CH2-ID1)



# swift

## variables

```
let maximumNumberOfLoginAttempts = 10
var currentLoginAttempt = 0
```

not mutable

mutable

```
let pi = 3.14159
// pi is inferred to be of type Double
let three = 3
let pointOneFourOneFiveNine = 0.14159
let pi = Double(three) + pointOneFourOneFiveNine
// pi equals 3.14159, and is inferred to be of type Double
let meaningOfLife = 42
// meaningOfLife is inferred to be of type Int
```

and then there is this...

```
let orangesAreOrange = true
let turnipsAreDelicious = false
```

```
let π = 3.14159
let 你好 = "你好世界"
let 🐶🐮 = "dogcow"
```

```
var friendlyWelcome = "Hello World!"
var friendlyWelcome: String = "Hello World!"

print(friendlyWelcome)

print("The current value of friendlyWelcome is \(friendlyWelcome)")
```

no need to set

see this: [https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift\\_Programming\\_Language/TheBasics.html#//apple\\_ref/doc/uid/TP40014097-CH5-ID309](https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/TheBasics.html#//apple_ref/doc/uid/TP40014097-CH5-ID309)

# Swift

## tuples

```
let http404Error = (404, "Not Found")  
// http404Error is of type (Int, String), and equals (404, "Not Found")
```

```
let (statusCode, statusMessage) = http404Error  
print("The status code is \(statusCode)")  
// prints "The status code is 404"  
print("The status message is \(statusMessage)")  
// prints "The status message is Not Found"
```

```
print("The status code is \(http404Error.0)")  
// prints "The status code is 404"  
print("The status message is \(http404Error.1)")  
// prints "The status message is Not Found"
```

```
let http200Status = (statusCode: 200, description: "OK")
```

```
print("The status code is \(http200Status.statusCode)")  
// prints "The status code is 200"  
print("The status message is \(http200Status.description)")  
// prints "The status message is OK"
```

see this: [https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift\\_Programming\\_Language/TheBasics.html#//apple\\_ref/doc/uid/TP40014097-CH5-ID309](https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/TheBasics.html#//apple_ref/doc/uid/TP40014097-CH5-ID309)

# swift

## optionals

```
let possibleNumber = "123"  
let convertedNumber = possibleNumber.toInt()  
// convertedNumber is inferred to be of type "Int?", or "optional Int"
```

```
var serverResponseCode: Int? = 404  
// serverResponseCode contains an actual Int value of 404  
serverResponseCode = nil  
// serverResponseCode now contains no value
```

can now set to nil :)

```
var surveyAnswer: String?  
// surveyAnswer is automatically set to nil
```

```
if convertedNumber != nil {  
    print("convertedNumber has an integer value of \(convertedNumber!).")  
}  
// prints "convertedNumber has an integer value of 123."
```

```
if let actualNumber = possibleNumber.toInt() {  
    print("\(possibleNumber)' has an integer value of \(actualNumber)")  
} else {  
    print("\(possibleNumber)' could not be converted to an integer")  
}  
// prints "'123' has an integer value of 123"
```

see this: [https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift\\_Programming\\_Language/TheBasics.html#//apple\\_ref/doc/uid/TP40014097-CH5-ID309](https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/TheBasics.html#//apple_ref/doc/uid/TP40014097-CH5-ID309)



# swift

## accessing optionals

optional

```
let possibleString: String? = "An optional string."  
let forcedString: String = possibleString! // requires an exclamation mark
```

! **unwrap** output to be **string**. Else: **error**

implicit unwrap

```
let assumedString: String! = "An implicitly unwrapped optional string."  
let implicitString: String = assumedString // no need for an exclamation mark
```

output always unwrapped to be **string**. Else: **error**

```
if assumedString != nil {  
    print(assumedString)  
}  
// prints "An implicitly unwrapped optional string."
```

```
if let definiteString = assumedString {  
    print(definiteString)  
}  
// prints "An implicitly unwrapped optional string."
```

Optional unwrapping is not my favorite part of swift

see this: [https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift\\_Programming\\_Language/TheBasics.html#//apple\\_ref/doc/uid/TP40014097-CH5-ID309](https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/TheBasics.html#//apple_ref/doc/uid/TP40014097-CH5-ID309)

# swift

## arrays

```
var shoppingList = ["Eggs", "Milk"]
```

```
print("The shopping list contains \(shoppingList.count) items.")  
// prints "The shopping list contains 2 items."
```

```
if shoppingList.isEmpty {  
    print("The shopping list is empty.")  
} else {  
    print("The shopping list is not empty.")  
}  
// prints "The shopping list is not empty."
```

```
shoppingList += ["Baking Powder"]  
shoppingList += ["Chocolate Spread", "Cheese", "Butter"]
```

```
var firstItem = shoppingList[0]  
// firstItem is equal to "Eggs"
```

```
shoppingList[0] = "Six eggs"
```

like a dequeue

```
let butter = shoppingList.removeLast()  
let sixEggs = shoppingList.removeAtIndex(0)
```

like a pop

see this: [https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift\\_Programming\\_Language/TheBasics.html](https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/TheBasics.html)

# swift

## dictionaries (skip)

```
var airports = ["YYZ": "Toronto Pearson", "DUB": "Dublin"]  
  
airports["LHR"] = "London"  
// the airports dictionary now contains 3 items
```

```
if let oldValue = airports.updateValue("Dublin Airport", forKey: "DUB") {  
    print("The old value for DUB was \(oldValue).")  
}  
// prints "The old value for DUB was Dublin."
```

```
airports["APL"] = "Apple International"  
// "Apple International" is not the real airport for APL, so delete it  
airports["APL"] = nil  
// APL has now been removed from the dictionary
```

```
let airportCodes = [String](airports.keys)  
// airportCodes is ["YYZ", "LHR"]  
  
let airportNames = [String](airports.values)  
// airportNames is ["Toronto Pearson", "London Heathrow"]
```

see this: [https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift\\_Programming\\_Language/TheBasics.html](https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/TheBasics.html)

# Swift

## loops

```
for index in 1...3 {  
    print("\(index) times 5 is \(index * 5)")  
}  
// 1 times 5 is 5  
// 2 times 5 is 10  
// 3 times 5 is 15
```

```
let names = ["Anna", "Alex", "Brian"]  
for name in names {  
    print("Hello, \(name)!")  
}  
// Hello, Anna!  
// Hello, Alex!  
// Hello, Brian!
```

```
for (index, value) in enumerate(names) {  
    println("Item \(index + 1): \(value)")  
}  
// Item 1: Anna  
// Item 2: Alex  
// Item 3: Brian
```

```
let numberOfLegs = ["spider": 8, "ant": 6, "cat": 4]  
for (animalName, legCount) in numberOfLegs {  
    print("\(animalName)s have \(legCount) legs")  
}  
// ants have 6 legs  
// cats have 4 legs  
// spiders have 8 legs
```

see this: [https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift\\_Programming\\_Language/TheBasics.html](https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/TheBasics.html)

# swift

## switch

```
let someCharacter: Character = "e"
switch someCharacter {
case "a", "e", "i", "o", "u":
    print("\(someCharacter) is a vowel")
case "b", "c", "d", "f", "g", "h", "j", "k", "l", "m",
     "n", "p", "q", "r", "s", "t", "v", "w", "x", "y", "z":
    print("\(someCharacter) is a consonant")
default:
    print("\(someCharacter) is not a vowel or a consonant")
}
// prints "e is a vowel"
```

no pass through

```
let somePoint = (1, 1)
switch somePoint {
case (0, 0):
    print("(0, 0) is at the origin")
case (_, 0):
    print("\(somePoint.0), 0) is on the x-axis")
case (0, _):
    print("0, \(somePoint.1) is on the y-axis")
case (-2...2, -2...2):
    print("\(somePoint.0), \(somePoint.1) is inside the box")
default:
    print("\(somePoint.0), \(somePoint.1) is outside of the box")
}
// prints "(1, 1) is inside the box"
```

"any" value

see this: [https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift\\_Programming\\_Language/TheBasics.html](https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/TheBasics.html)

# Swift

## switch continued...

```
let anotherPoint = (2, 0)
switch anotherPoint {
case (let x, 0):
    print("on the x-axis with an x value of \(x)")
case (0, let y):
    print("on the y-axis with a y value of \(y)")
case let (x, y):
    print("somewhere else at (\(x), \(y))")
}
// prints "on the x-axis with an x value of 2"
```

"any" value and set

```
let yetAnotherPoint = (1, -1)
switch yetAnotherPoint {
case let (x, y) where x == y:
    print("\(x), \(y) is on the line x == y")
case let (x, y) where x == -y:
    print("\(x), \(y) is on the line x == -y")
case let (x, y):
    print("\(x), \(y) is just some arbitrary point")
}
// prints "(1, -1) is on the line x == -y"
```

very powerful, concise,  
readable

see this: [https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift\\_Programming\\_Language/TheBasics.html](https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/TheBasics.html)



# swift

## functions

internal name

input type

```
func sayHello(personName: String ) -> String {  
    let greeting = "Hello, " + personName + "!"  
    return greeting  
}
```

return type

internal name

input type

external name

```
func join(string s1: String, toString s2: String, withJoiner joiner: String)  
    -> String {  
    return s1 + joiner + s2  
}
```

return type

external name

passed value

```
join(string: "hello", toString: "world", withJoiner: ", ")  
// returns "hello, world"
```

see this: [https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift\\_Programming\\_Language/TheBasics.html](https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/TheBasics.html)

There are too many ways of defining functions to cover it all. For instance you can also setup default values...

array of ints

```
func minMax(array: [ Int ]) -> (min: Int , max: Int ) {  
    var currentMin = array[0]  
    var currentMax = array[0]  
    for value in array[1..  
        array.count] {  
        if value < currentMin {  
            currentMin = value  
        } else if value > currentMax {  
            currentMax = value  
        }  
    }  
    return (currentMin, currentMax)  
}
```

return tuple

tuple keys are external names!!

```
let bounds = minMax([8, -6, 2, 109, 3, 71])  
print("min is \(bounds.min) and max is \(bounds.max)")  
// prints "min is -6 and max is 109"
```

see this: [https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift\\_Programming\\_Language/TheBasics.html](https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/TheBasics.html)

# Swift

## classes and properties

```
class DataImporter {  
    var fileName = "data.txt"  
    // the DataImporter class would provide data importing functionality here  
}
```

class variable

```
class DataManager {  
    lazy var importer = DataImporter()  
    var data = [String]()  
    // the DataManager class would provide data management functionality here  
}
```

lazy instantiation

```
let manager = DataManager()  
manager.data.append("Some data")  
manager.data.append("Some more data")  
// the DataImporter instance for the importer property has not yet been created
```

class initialized, but importer is not set

```
println(manager.importer.fileName)  
// the DataImporter instance for the importer property has now been created  
// prints "data.txt"
```

when accessed first, sets value

see this: [https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift\\_Programming\\_Language/TheBasics.html](https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/TheBasics.html)

# Swift

## classes and properties

```
class StepCounter {  
    var totalSteps: Int = 0 {  
        willSet(newTotalSteps) {  
            println("About to set totalSteps to \(newTotalSteps)")  
        }  
        didSet {  
            if totalSteps > oldValue {  
                println("Added \(totalSteps - oldValue) steps")  
            }  
        }  
    }  
}
```

no custom setter or getters

but we can still do custom actions around the property access

```
let stepCounter = StepCounter()  
stepCounter.totalSteps = 200  
// About to set totalSteps to 200  
// Added 200 steps  
stepCounter.totalSteps = 360  
// About to set totalSteps to 360  
// Added 160 steps  
stepCounter.totalSteps = 896  
// About to set totalSteps to 896  
// Added 536 steps
```

see this: [https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift\\_Programming\\_Language/TheBasics.html](https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/TheBasics.html)

```
class Counter {  
    var count = 0  
    func increment() {  
        count++  
    }  
    func incrementBy(amount: Int) {  
        count += amount  
    }  
    func reset() {  
        count = 0  
    }  
}
```

```
class Counter {  
    var count: Int = 0  
    func incrementBy(amount: Int, numberOfTimes: Int) {  
        count += amount * numberOfTimes  
    }  
}
```

see this: [https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift\\_Programming\\_Language/TheBasics.html](https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/TheBasics.html)

# swift

## inheritance

```
class NamedShape {
    var numberOfSides: Int = 0
    var name: String

    init(name: String) {
        self.name = name
    }

    func simpleDescription() -> String {
        return "A shape with \(numberOfSides) sides."
    }
}
```

```
class Square: NamedShape {
    var sideLength: Double

    init(sideLength: Double, name: String) {
        self.sideLength = sideLength
        super.init(name: name)
        numberOfSides = 4
    }

    func area() -> Double {
        return sideLength * sideLength
    }

    override func simpleDescription() -> String {
        return "A square with sides of length \(sideLength)."
    }
}
```



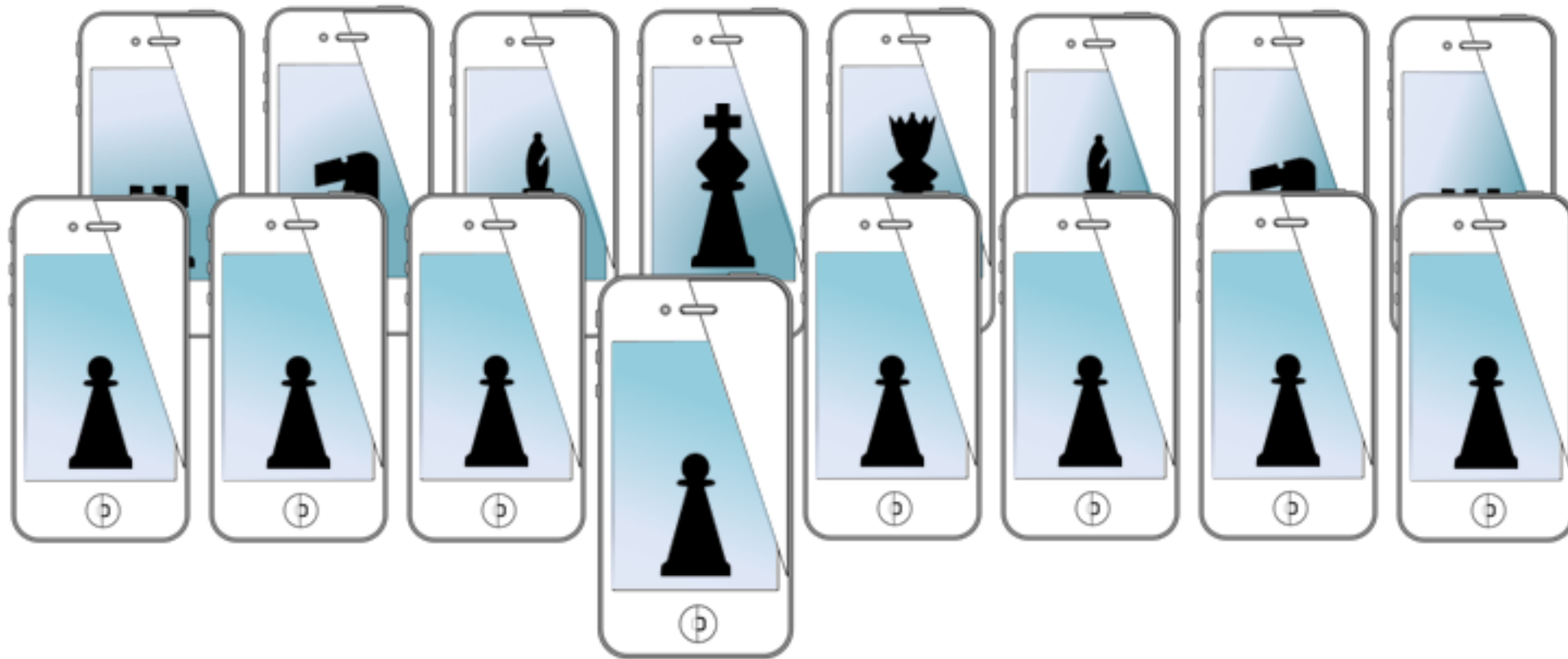
see this: [https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift\\_Programming\\_Language/TheBasics.html](https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/TheBasics.html)

Lots more on the inter-webs!

Need more help on MVC's ? Check out Ray Wenderlich:

<http://www.raywenderlich.com/46988/ios-design-patterns>

# MOBILE SENSING & LEARNING



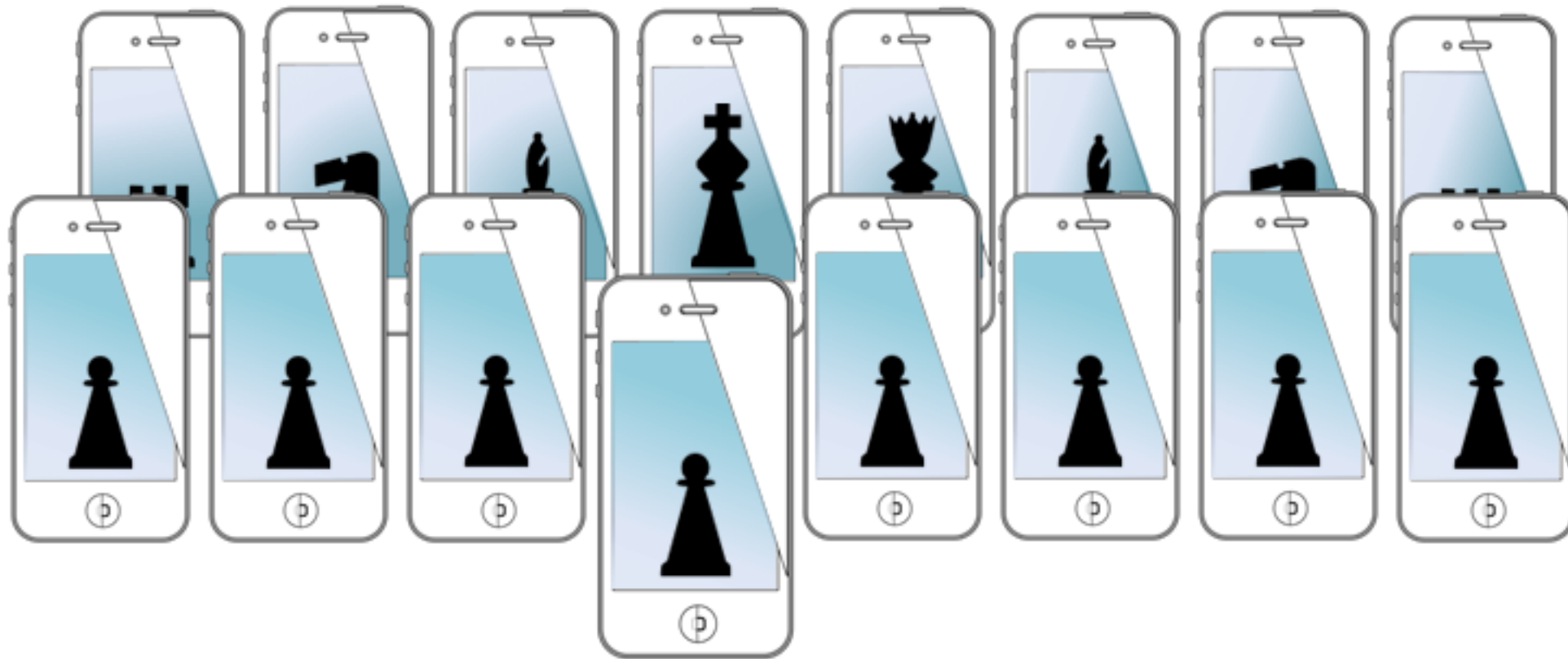
## CSE5323 & 7323

Mobile Sensing & Learning

week two, lecture three: UI elements and swift

Eric C. Larson, Lyle School of Engineering,  
Computer Science and Engineering, Southern Methodist University

# MOBILE SENSING & LEARNING



## CSE5323 & 7323

Mobile Sensing & Learning

Supplemental Slides: page view controllers,  
programmatic layout, and core data

Eric C. Larson, Lyle School of Engineering,  
Computer Science and Engineering, Southern Methodist University

# Supplemental Slides

- we do not explicitly cover these topics in class anymore
- some of the info in these slides may be deprecated!
- otherwise, have fun browsing the material

**disclaimer!**

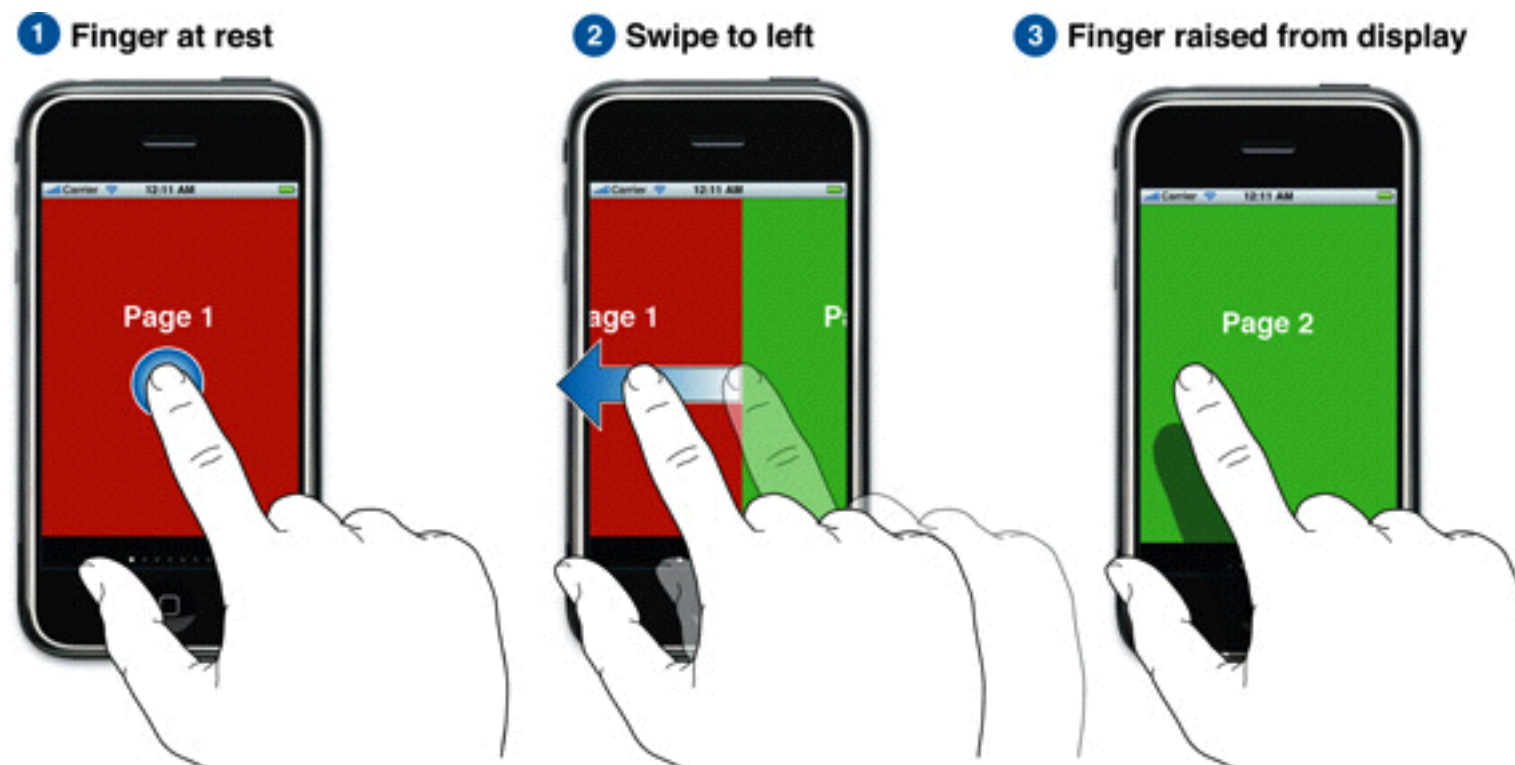
# page view controller

- place UINavigationController in storyboard
- place a “root controller” for the page
  - adopt <UINavigationControllerDataSource>
  - instantiate pageViewController
  - instantiate views to be paged

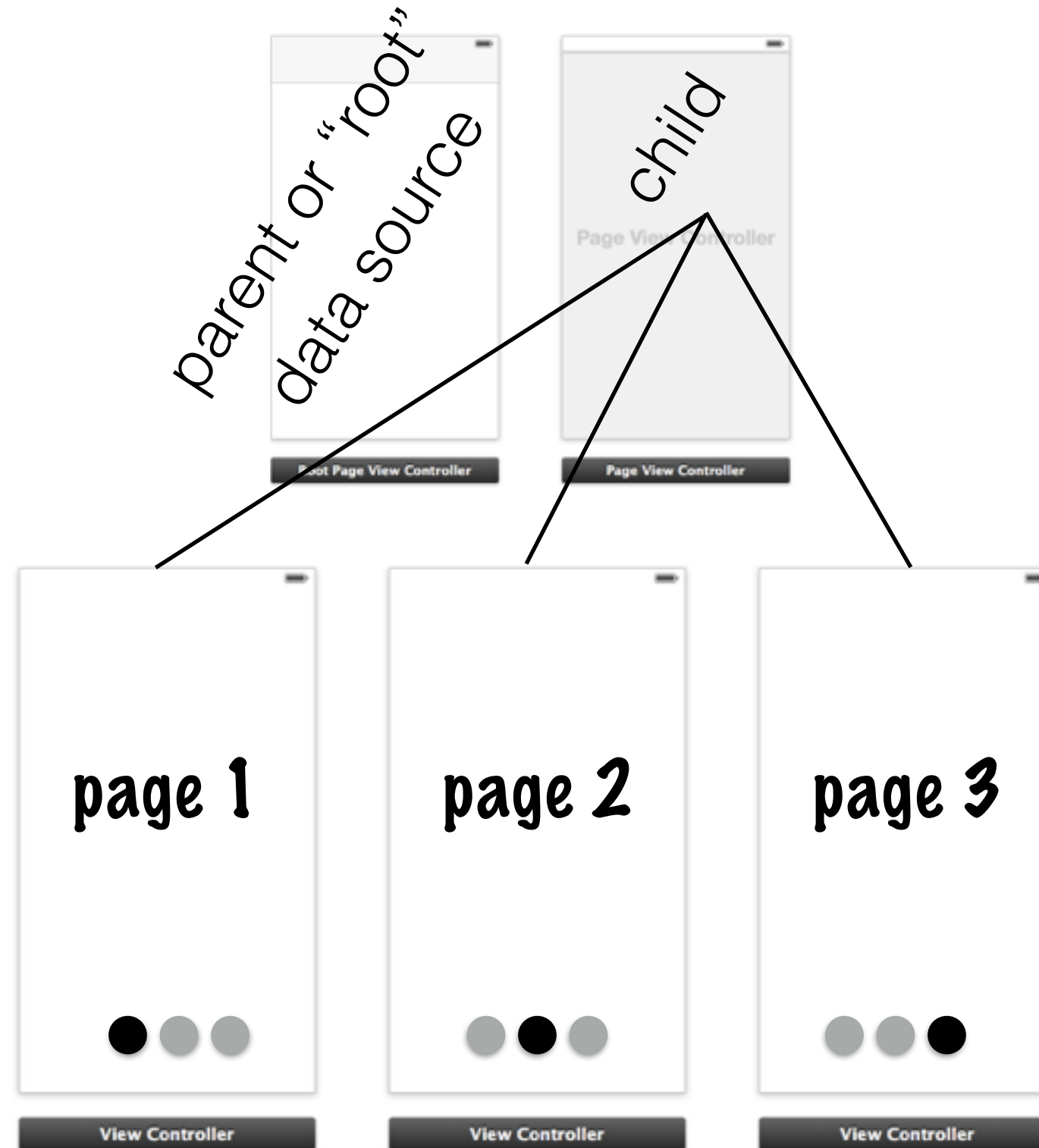


# page view controller

- place `UIPageViewController` in storyboard
- place a “root controller” for the page
  - adopt `<UIPageViewControllerDataSource>`
  - instantiate `pageViewController` from “root”
  - instantiate views to be paged in “root”



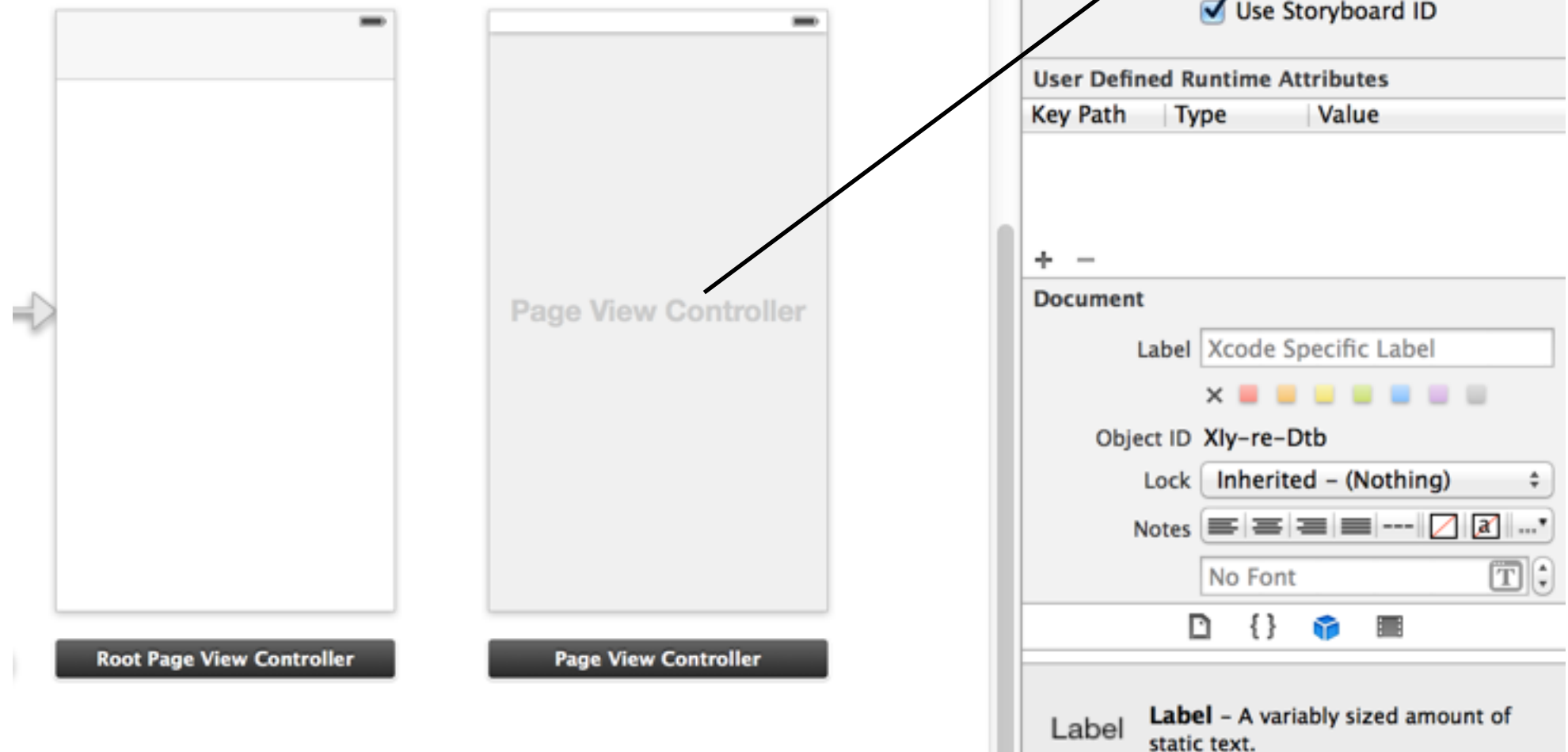
# page view controller



different instantiations of view controller

# page view controller

no need to subclass the page controller!



but root of the page controller must be the data source...

# root page view controller

instantiation in root view controller

```
@property (strong, nonatomic) UIPageViewController * pageViewController;  
@property (strong, nonatomic) NSArray *pageContent;  
  
_pageViewController = [self.storyboard instantiateViewControllerWithIdentifier:@"PageViewController"];  
_pageViewController.dataSource = self;
```

set first page

instantiate!

in viewDidLoad

```
[self.pageViewController setViewControllers:firstPageToDisplay // the page is a view controller!  
                        direction:UIPageViewControllerNavigationDirectionForward  
                        animated:NO  
                        completion:nil];
```

```
[self addChildViewController:_pageViewController];  
[self.view addSubview:_pageViewController.view];  
[self.pageViewController didMoveToParentViewController:self];
```

apple says do  
this, in order

some datasource protocol methods

```
- (NSInteger)presentationCountForPageViewController:(UIPageViewController *)pageViewController  
{  
    return [self.pageContent count];  
}  
  
- (NSInteger)presentationIndexForPageViewController:(UIPageViewController *)pageViewController  
{  
    return 0;  
}
```

# root page view controller

some datasource protocol methods (cont.)

```
- (NSInteger)presentationCountForPageViewController:(UIPageViewController *)pageViewController
{
    return [self.pageContent count];
}

- (NSInteger)presentationIndexForPageViewController:(UIPageViewController *)pageViewController
{
    return 0;
}

-(UIViewController*)pageViewController:(UIPageViewController *)pageViewController
viewControllerBeforeViewController:(UIViewController *)viewController
{}

-(UIViewController*)pageViewController:(UIPageViewController *)pageViewController
viewControllerAfterViewController:(UIViewController *)viewController
{}
```

1. create pages (VCs)
2. set any information for loading
3. return the instantiated VC

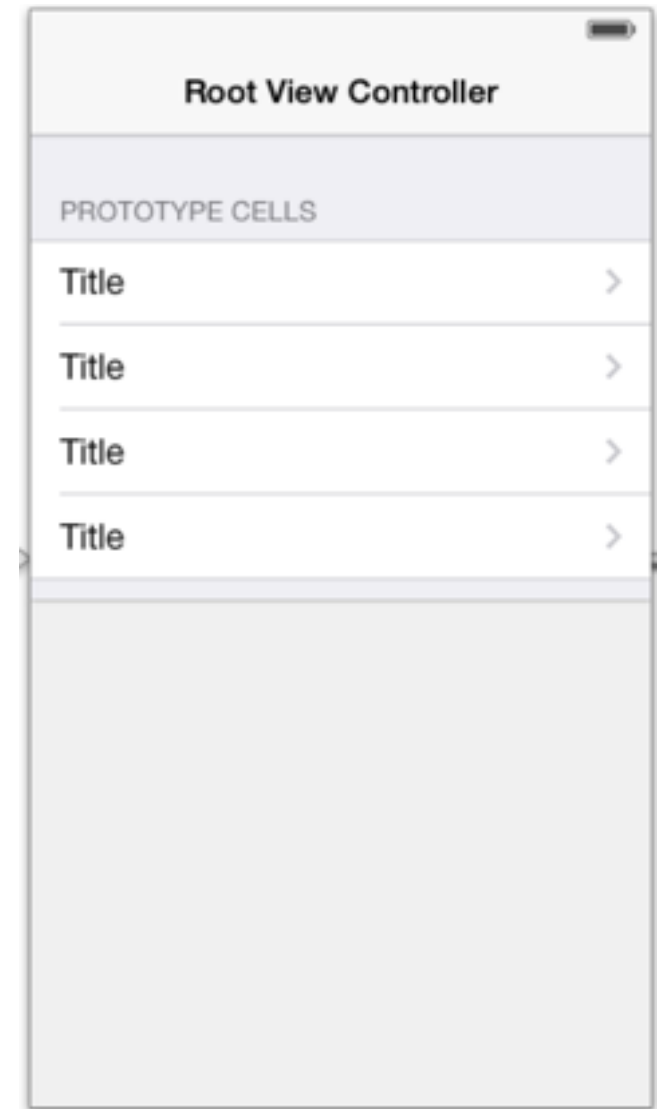


# page view demo



# assignment one

- Automatic Layout (storyboard and programmatically)
- UIButtons (created in storyboard and programmatically)
- Sliders (created in storyboard and programmatically)
- Labels (created in storyboard and programmatically)
- Stepper
- Switch
- Picker (Date or otherwise)
- UINavigationController
- **UISegmentedControl**
- **NSTimer** (which should repeat and somehow update the UIView)
- UIScrollView (with scrollable, zoomable content)
- UIPageViewController
- UIImageView
- **(optional) Persistent storage via CoreData**



due Friday, Feb. 7

# programmatic UI creation

```
@property (strong, nonatomic) IBOutlet UIButton *button;
```

```
...
```

```
// a button, created programmatically
self.button = [UIButton buttonWithType:UIButtonTypeSystem];

// set a target method for a control event, touch down
[self.button addTarget:self
                  action:@selector(updateLabelFromProgramButton:)
                  forControlEvents:UIControlEventTouchUpInside];

// set the button attribute
[self.button setTitle:@"PButton" forState:UIControlStateNormal];
```

# visual format language

```
// say that these exist and are initialized and added to the view as subviews
```

```
UIButton *button;
```

```
UILabel *label;
```

```
[button setTranslatesAutoresizingMaskIntoConstraints:NO];
```

```
// setup button and label constraints, also make same size
```

```
NSDictionary *varBindings = NSDictionaryOfVariableBindings(button, label);
```

```
NSArray *constraints =
```

```
    [NSLayoutConstraint constraintsWithVisualFormat:@"|-[button]-24-[label(==button)]-|"
```

```
        options:0
```

```
        metrics:nil
```

```
        views:varBindings];
```

```
[self.view addConstraints:constraints];
```

same size as button

8 points from left side

24 points between

```
// metrics for use in visual constraints
```

```
NSDictionary *metrics = @{@"spacing":@"10.0"};
```

```
[NSLayoutConstraint constraintsWithVisualFormat:@"|-[button]-spacing-[label(==button)]-|"
```

```
        options:0
```

```
        metrics:metrics
```

```
        views:varBindings];
```

```
@“V:|-[button]”
```

```
options:NSLayoutFormatAlignAllTop | NSLayoutFormatAlignAllCenterX
```

# core data databases

- allows access to SQLite database
- integrated deeply into Xcode and into iOS
- highly optimized
- excellent for storing persistent table data
  - but usable for most anything

# core data schema

## ENTITIES

**E** Student

**E** Teams

## FETCH REQUESTS

## CONFIGURATIONS

**C** Default

### ▼ Attributes

Attribute ▲	Type	
<b>S</b> hardware	String	↕
<b>S</b> name	String	↕
+ -		

```
@interface Teams : NSObject
```

```
@property (nonatomic, retain) NSString * name;  
@property (nonatomic, retain) NSString * hardware;  
@property (nonatomic, retain) NSSet *members;
```

```
@end
```

## ENTITIES

**E** Student

**E** Teams

## FETCH REQUESTS

## CONFIGURATIONS

**C** Default

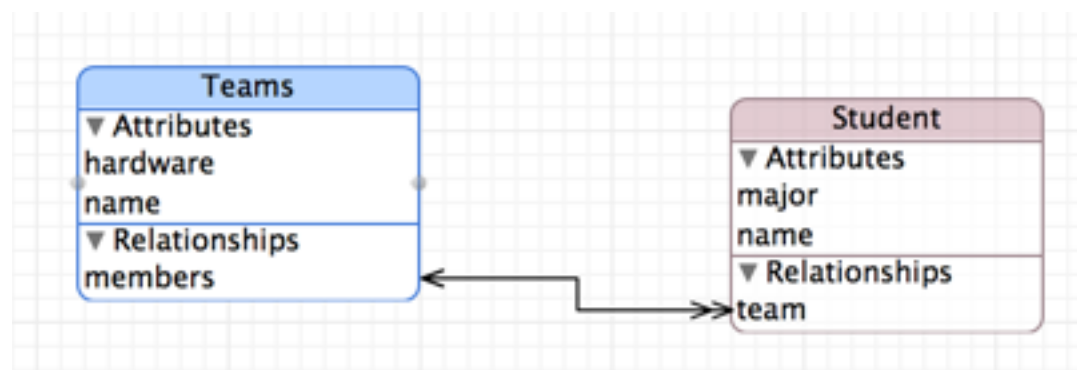
### ▼ Attributes

Attribute ▲	Type	
<b>S</b> major	String	↕
<b>S</b> name	String	↕
+ -		

```
@interface Student : NSObject
```







```
@property (nonatomic, retain) NSString * name;  
@property (nonatomic, retain) NSString * major;  
@property (nonatomic, retain) Teams *team;
```

```
@end
```





# core data

- schema creation  create SQLite Database on phone
- automatic subclassing  enable access through properties
- NSManagedObject  bundle “data models”
- NSManagedObjectContext  get “context” for using data model
- NSPersistentStore  coordinate access to the data model
- NSFetchRequest  create and execute queries

# core data setup

```
// Getter for managed context
- (NSManagedObjectContext *) managedObjectContext {

    if(!_managedObjectContext){
        // create the storage coordinator
        NSPersistentStoreCoordinator *coordinator = [self persistentStoreCoordinator];
        if (coordinator != nil) {
            _managedObjectContext = [[NSManagedObjectContext alloc] init];
            [_managedObjectContext setPersistentStoreCoordinator: coordinator];
        }
    }

    return _managedObjectContext;
}

// getter for the storage coordinator
- (NSPersistentStoreCoordinator *)persistentStoreCoordinator {
    if (!_persistentStoreCoordinator) {

        // this points to our model
        NSURL *storeUrl = [NSURL fileURLWithPath: [[self applicationDocumentsDirectory]
                                                    stringByAppendingPathComponent: @"modelName.sqlite"]];

        NSError *error = nil;
        _persistentStoreCoordinator = [[NSPersistentStoreCoordinator alloc]
                                         initWithManagedObjectModel:[self managedObjectModel]];

        if(![_persistentStoreCoordinator addPersistentStoreWithType:NSSQLiteStoreType
                                                             configuration:nil URL:storeUrl options:nil error:&error]) {
            // exit gracefully if you need the database to function in the UI
        }
    }
    return _persistentStoreCoordinator;
}
```

# core data setup

```
// getter for the storage coordinator
- (NSPersistentStoreCoordinator *)persistentStoreCoordinator {
    if (!_persistentStoreCoordinator) {

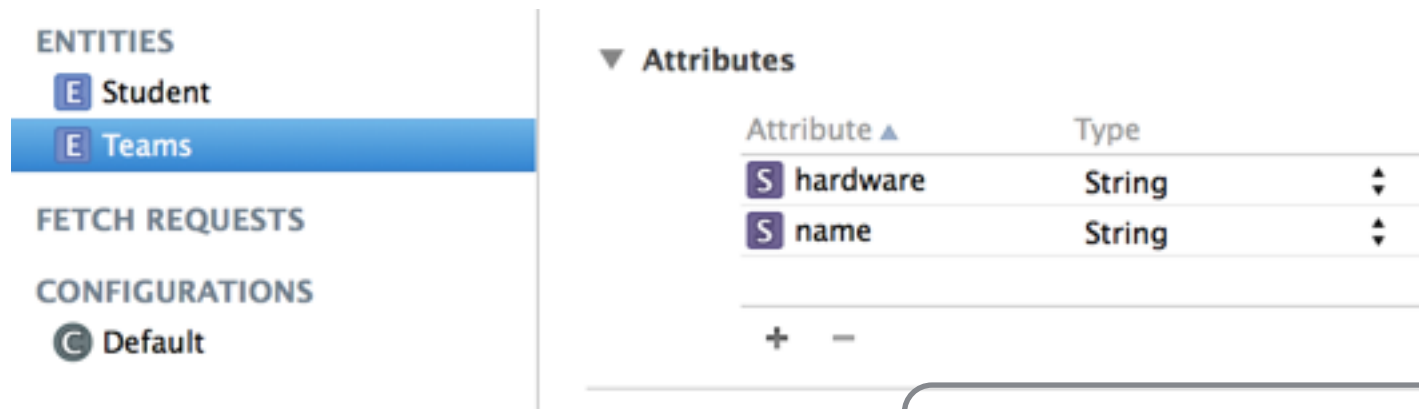
        // this points to our model
        NSURL *storeUrl = [NSURL fileURLWithPath: [[self applicationDocumentsDirectory]
                                                    stringByAppendingPathComponent: @"ModelName.sqlite"]];

        NSError *error = nil;
        _persistentStoreCoordinator = [[NSPersistentStoreCoordinator alloc]
                                        initWithManagedObjectModel:[self managedObjectModel]];

        if(![_persistentStoreCoordinator addPersistentStoreWithType:NSSQLiteStoreType
                    configuration:nil URL:storeUrl options:nil error:&error]) {
            // exit gracefully if you need the database to function in the UI
        }
    }
    return _persistentStoreCoordinator;
}

// getter for the object model, create if needed
- (NSManagedObjectModel *)managedObjectModel {
    if (!_managedObjectModel) {
        _managedObjectModel = [NSManagedObjectModel mergedModelFromBundles:nil];
    }
    return _managedObjectModel;
}
```

# entering data



create a new entity from model

```
// get a new entry
team = [NSEntityDescription insertNewObjectForEntityForName:@"Teams"
                                     inManagedObjectContext:self.managedObjectContext];

// save the attributes
team.name = self.teamNameTextField.text;
team.hardware = [self assignHardware];

// save into the database
NSError *error;
if (![self.managedObjectContext save:&error]) {
    NSLog(@"save database failed: %@", [error localizedDescription]);
}
```

set attributes

not saved in database until here

# queries in core data

```
-(NSArray*)getAllTeamsFromDatabase
{
    // initializing NSFetchRequest
    NSFetchRequest *fetchRequest = [[NSFetchRequest alloc] init];

    //Setting Entity to be Queried
    NSEntityDescription *entity = [NSEntityDescription entityForName:@"Teams"
                                inManagedObjectContext:self.managedObjectContext];

    [fetchRequest setEntity:entity];
    NSError* error;

    // Query on managedObjectContext With Generated fetchRequest
    NSArray *fetchedRecords = [self.managedObjectContext executeFetchRequest:fetchRequest error:&error];

    // Returning Fetched Records
    return fetchedRecords;
}
```

request

fetch

entity to request from

array of results, even if size=0

```
-(NSArray*)getTeamFromDatabase:(NSString*)teamName
{
    // initializing NSFetchRequest
    ...

    fetchRequest.predicate =
        [NSPredicate predicateWithFormat:@"name = %@", teamName];

    ...

    // Returning Fetched Records
    return [self.managedObjectContext executeFetchRequest:fetchRequest error:&error];
}
```

set predicate

@name = %@  
@name contains[c] %@  
@value > 7  
@team.name = %@  
@any student.name contains %@

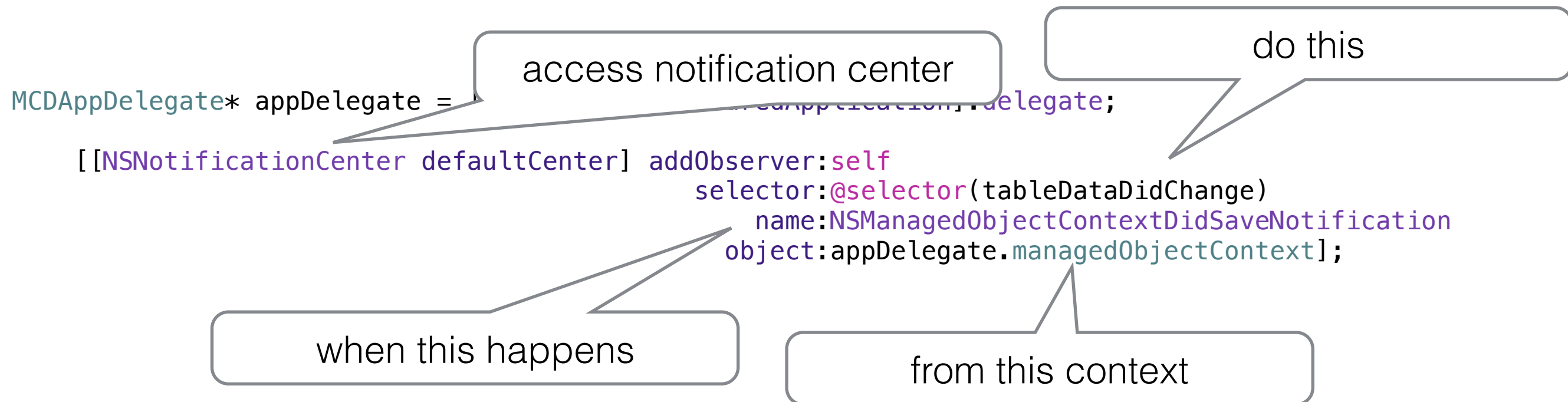
# core data demo

- Who Was In That!
- Class Teams! will make available on website



# notifications

- NotificationCenter - a radio station for which any method can tune in on



lets add notifications to WhoWasInThat!