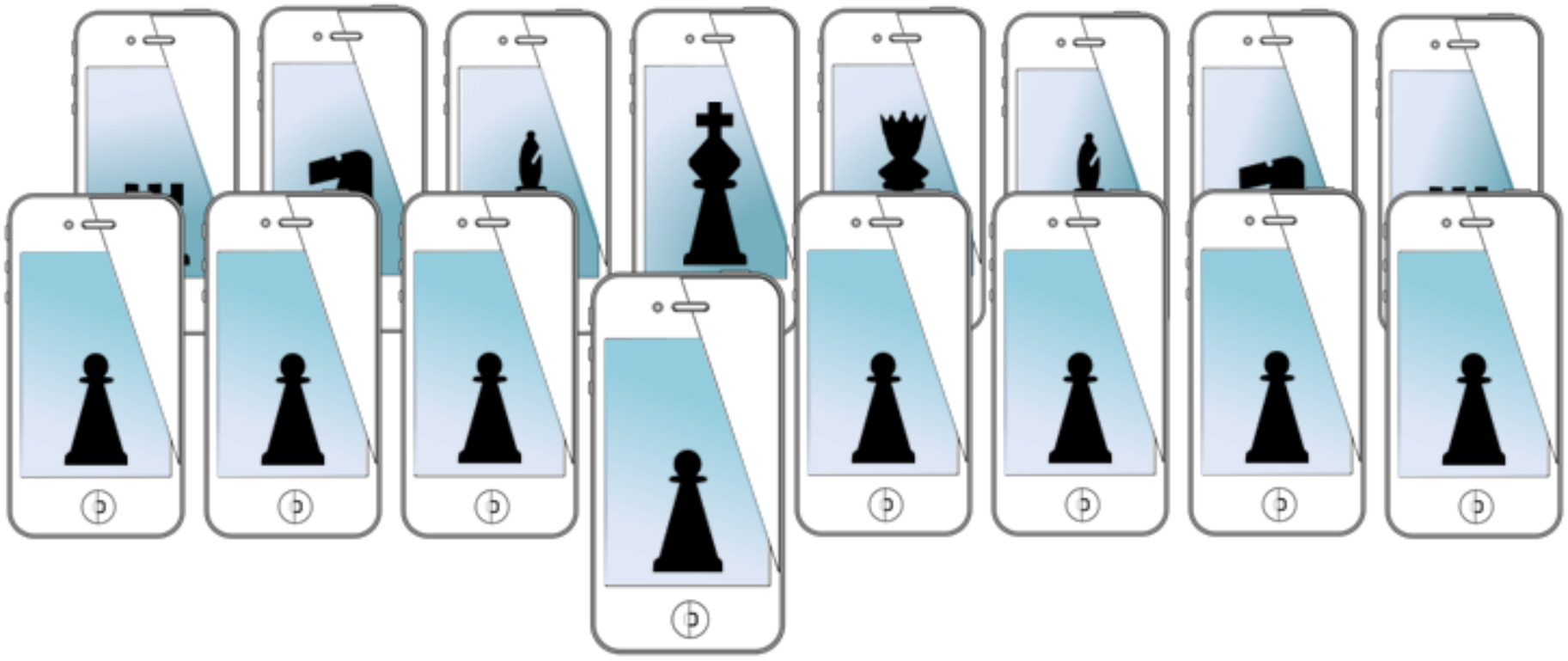
MOBILE SENSING & LEARNING



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
  - TableViewControllers, NavigationViewController, CollectionViewController, UIViewController
- storyboard (with UIViewController)
  - outlets, auto layout, programatic creation
  - timers, UIScrollView, image assets,

MVC's

controller has direct connection to view class

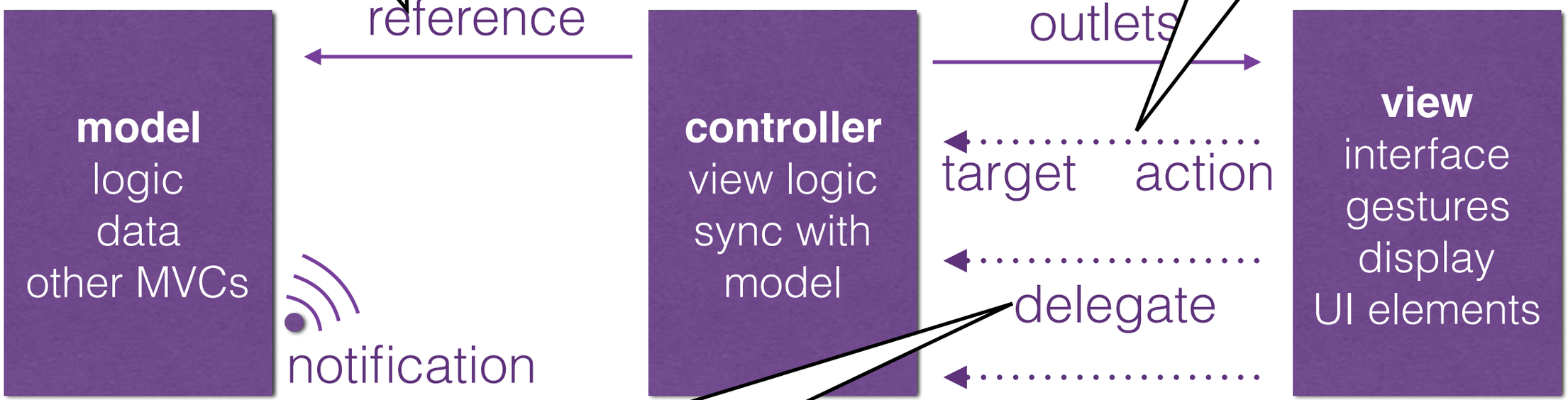
```
@property (weak, nonatomic) IBOutlet UITextField *firstName;
@property (weak, nonatomic) IBOutlet UITextField *lastName;
@property (weak, nonatomic) IBOutlet UITextField *phoneNumber;
```

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

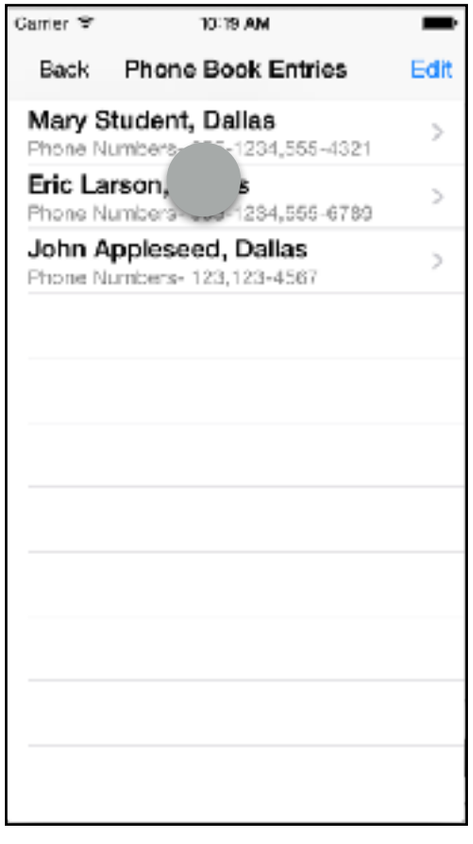
Legend

direct connection
indirect action
general broadcast

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

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

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

Destination Controller

view is unfrozen, property memory allocated

view outlets are ready for interaction

`viewDidLoad`

`viewWillAppear`

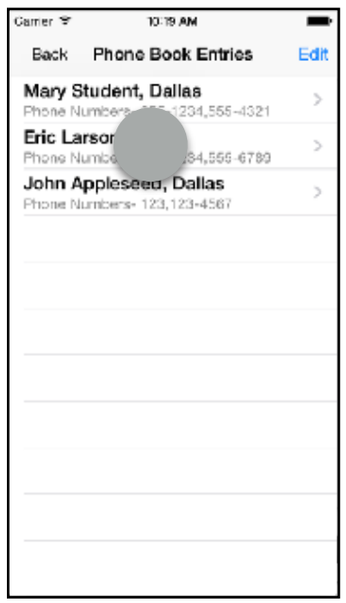
`viewDidAppear`

`viewWillDisappear`

`viewDidDisappear`

memory deallocated when app is ready

source



destination

user

the storyboard

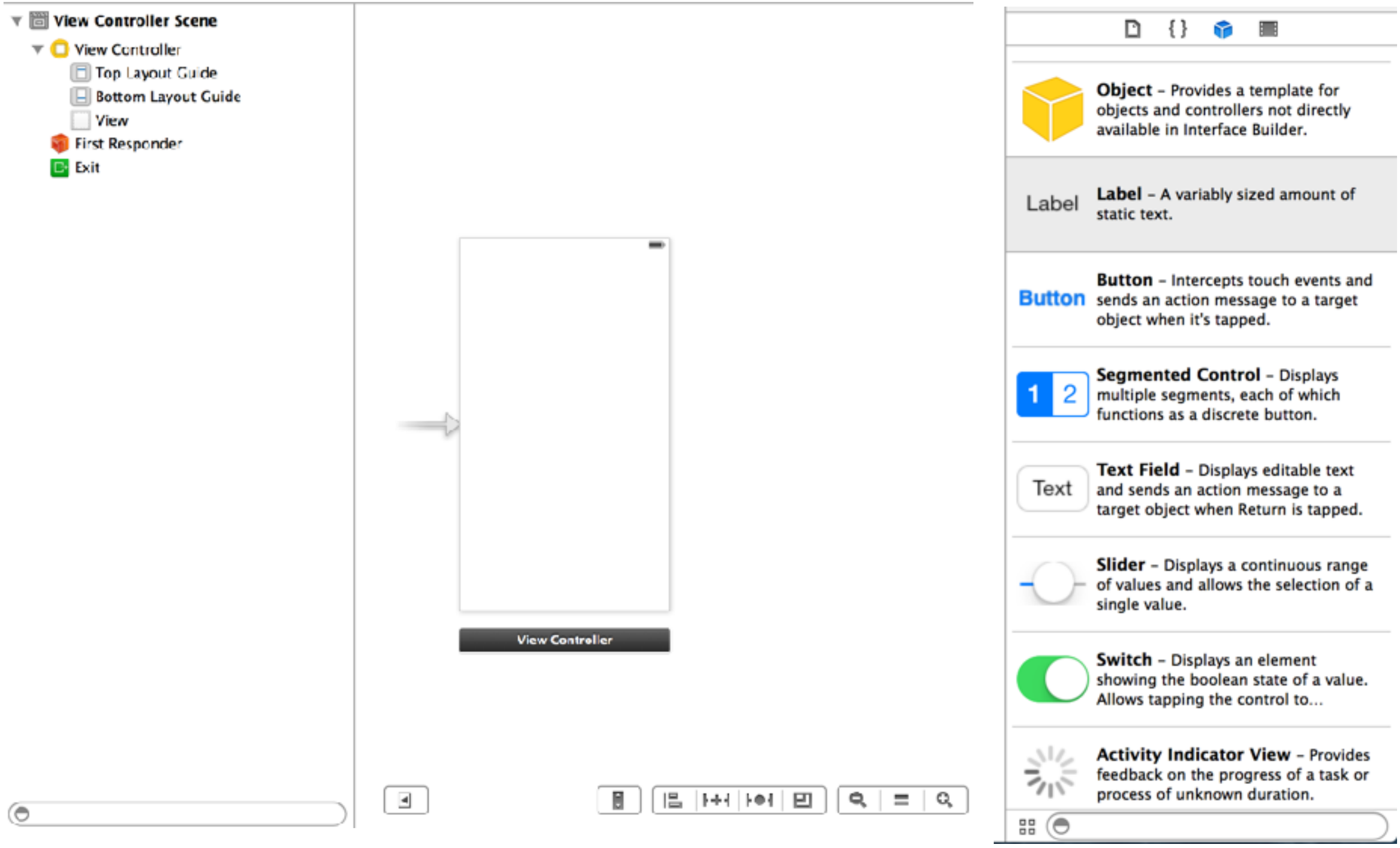
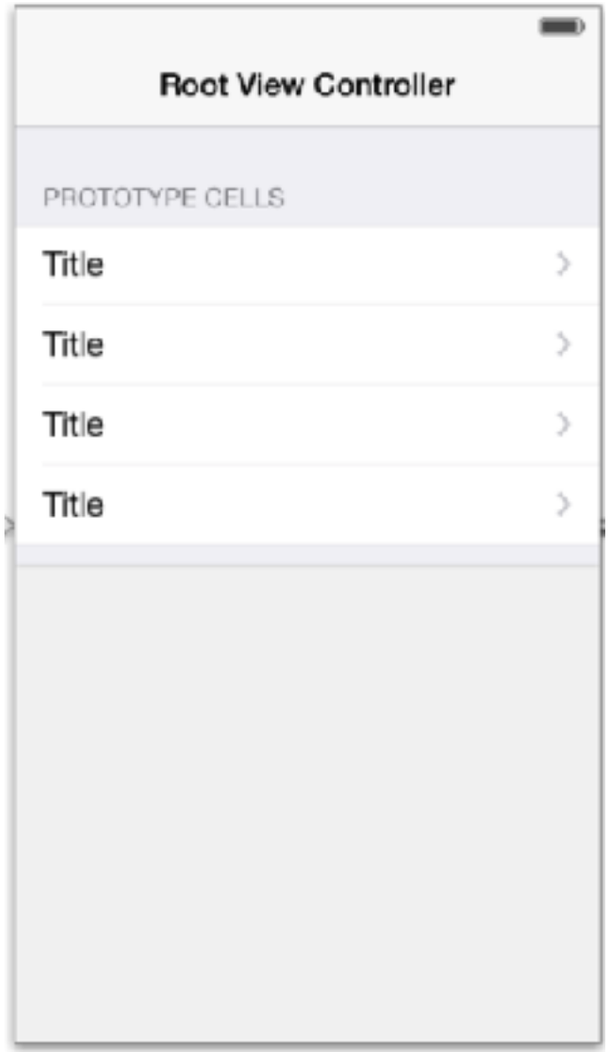
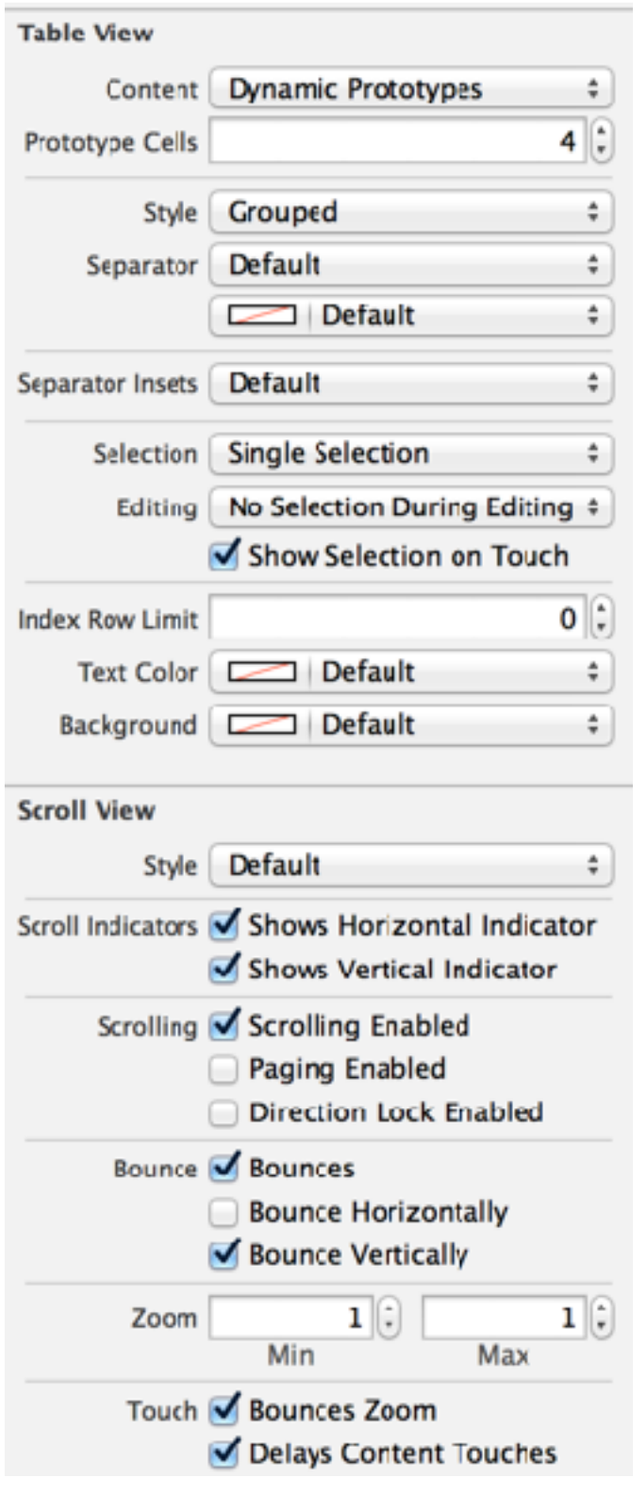
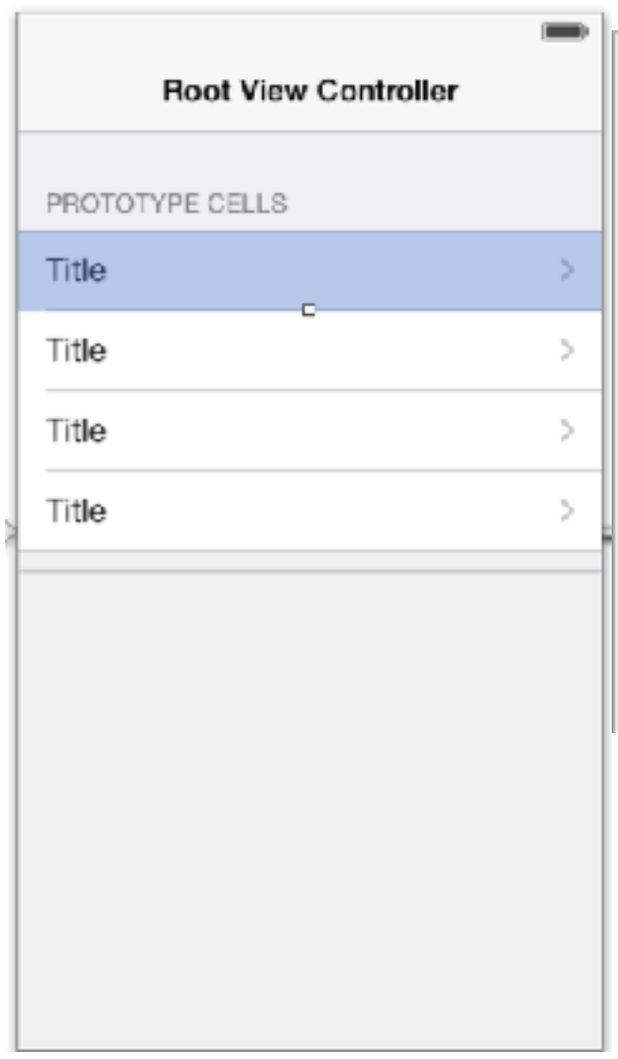


table view controller







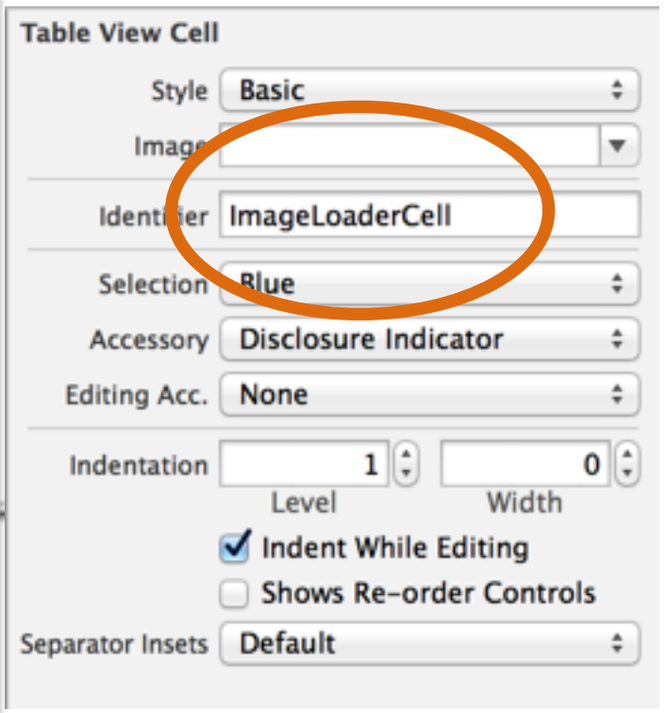


table view controller

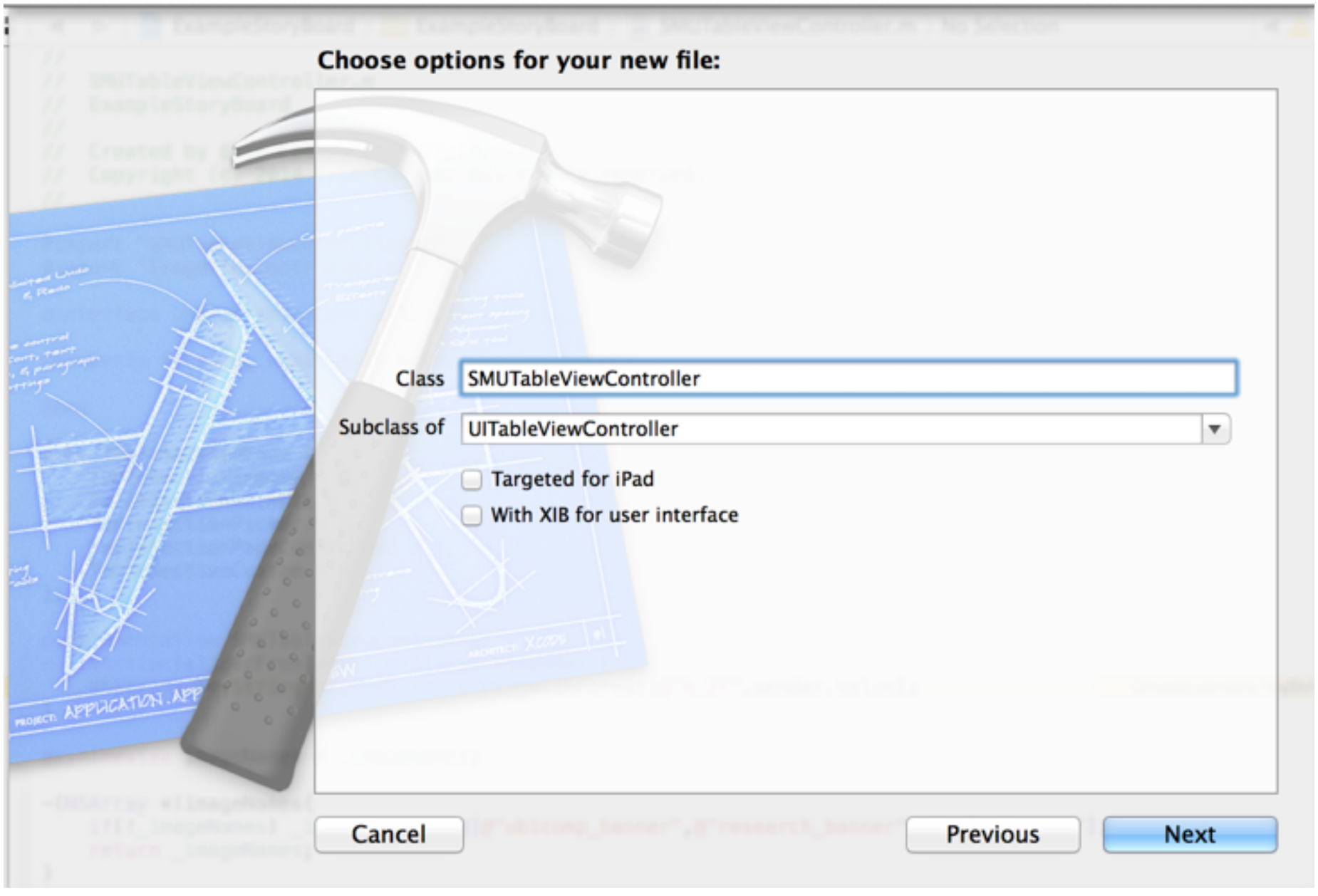


table view controller

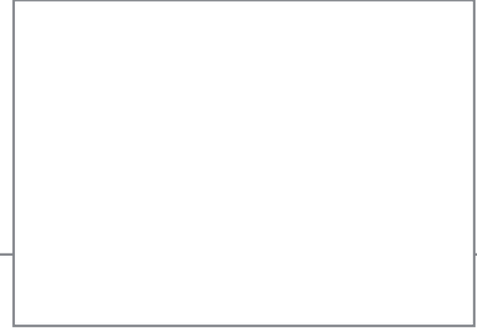
- must implement “data source” methods

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


- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(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;
}
```

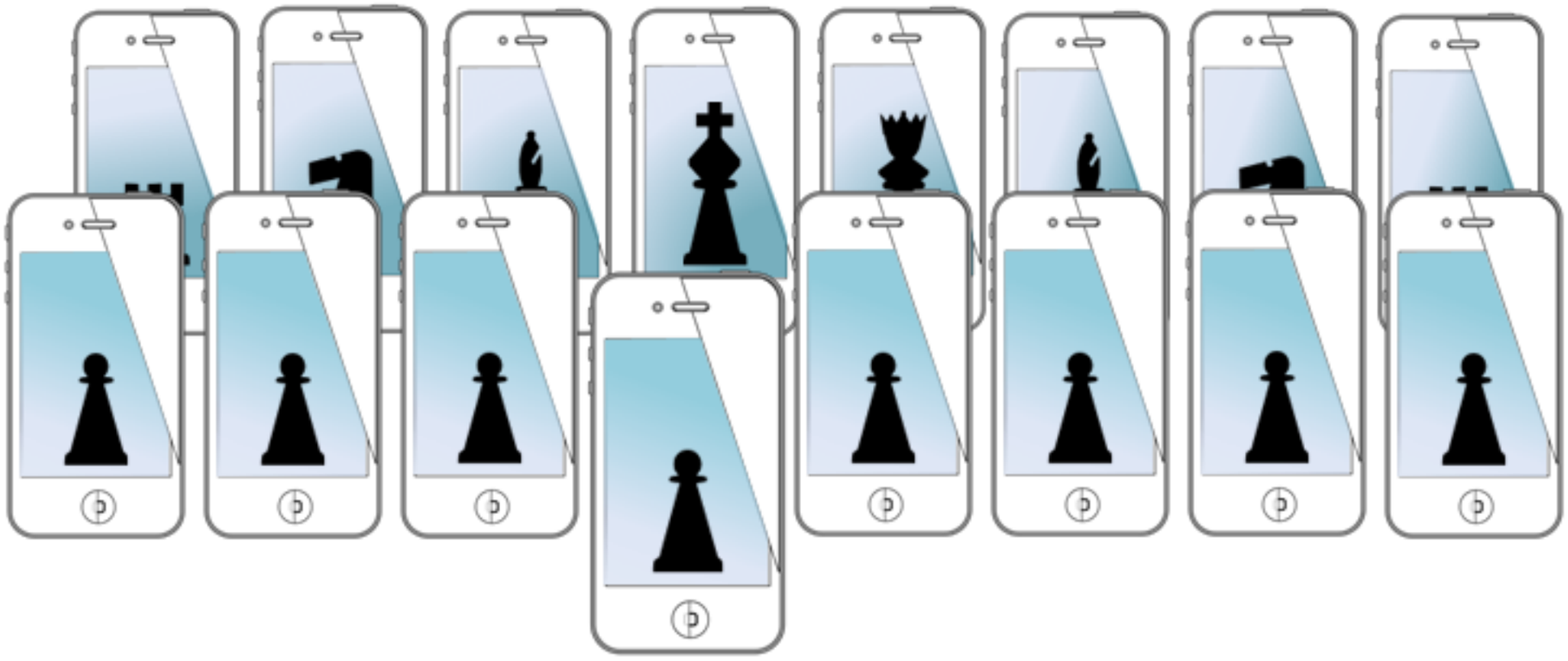
cell prototype from storyboard

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

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

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

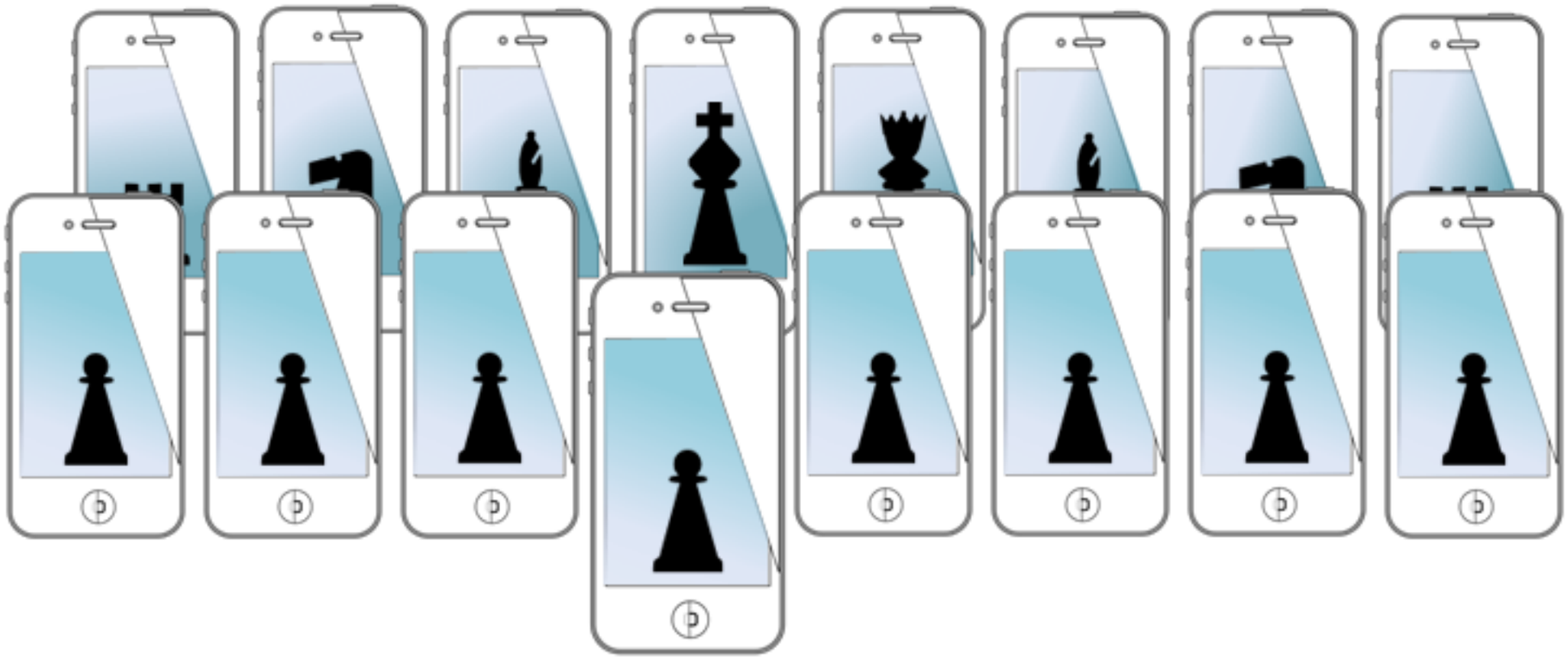
set VC as delegate

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

one of many methods in the protocol

demo

MOBILE SENSING & LEARNING



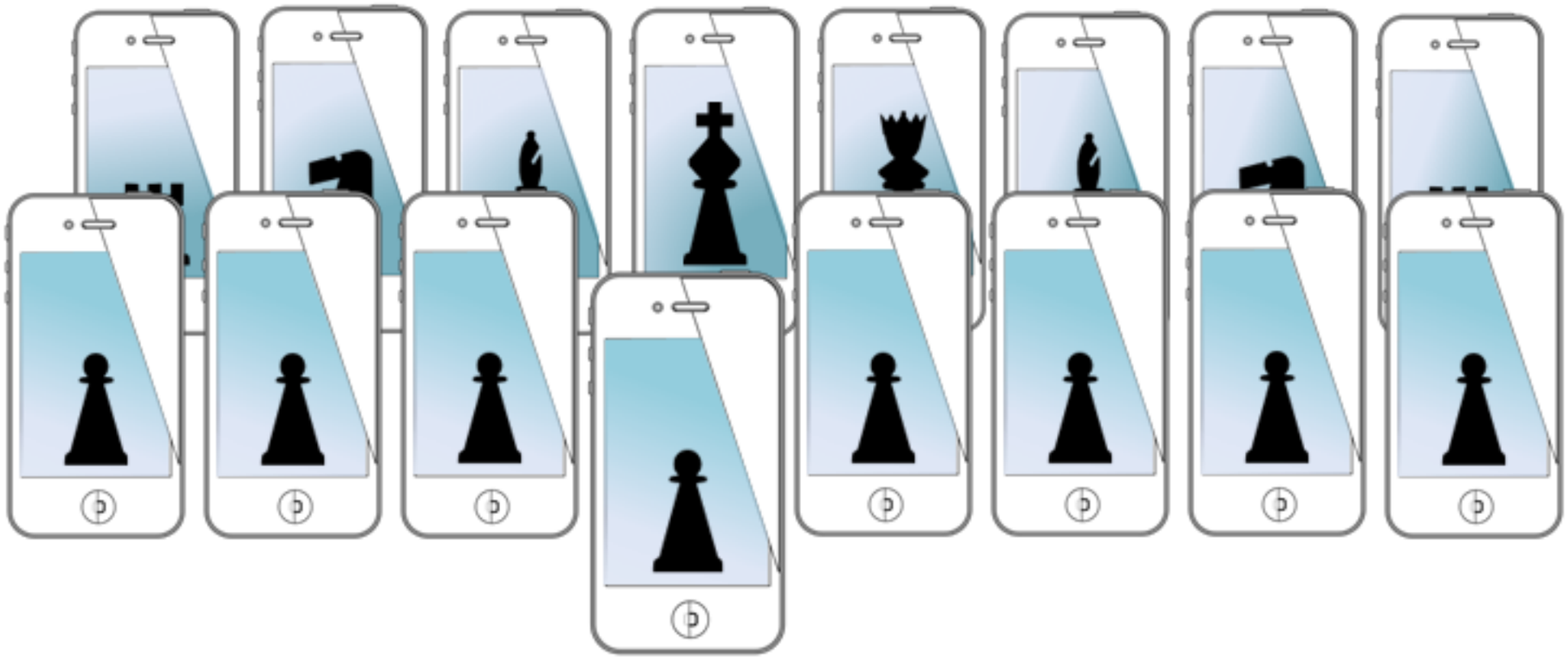
CSE5323 & 7323

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 two: UI elements swift

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

course logistics

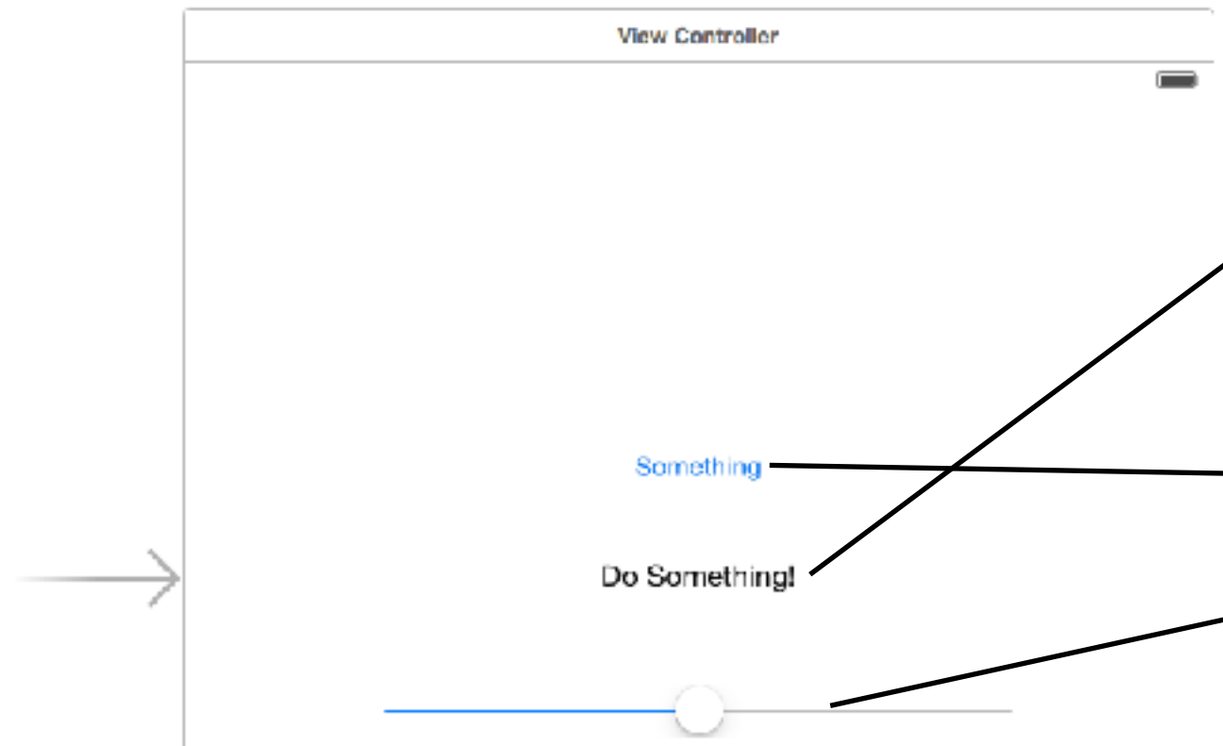
- **TA**: Xinyi Ding, Tu 3-7 in Caruth 384
- Reminder: University developer program!
  - Need your:
    - email that you want invite sent to
    - device Name (i.e., Eric's iPhone)
    - Phone SEID: Settings > general > about > SEID
- A1 due at **end of next week**
  - come show me the app in my office
  - or make a video of the app and submit it
  - questions?

agenda

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

target and action

- UI elements communicate back to their controllers with **actions**



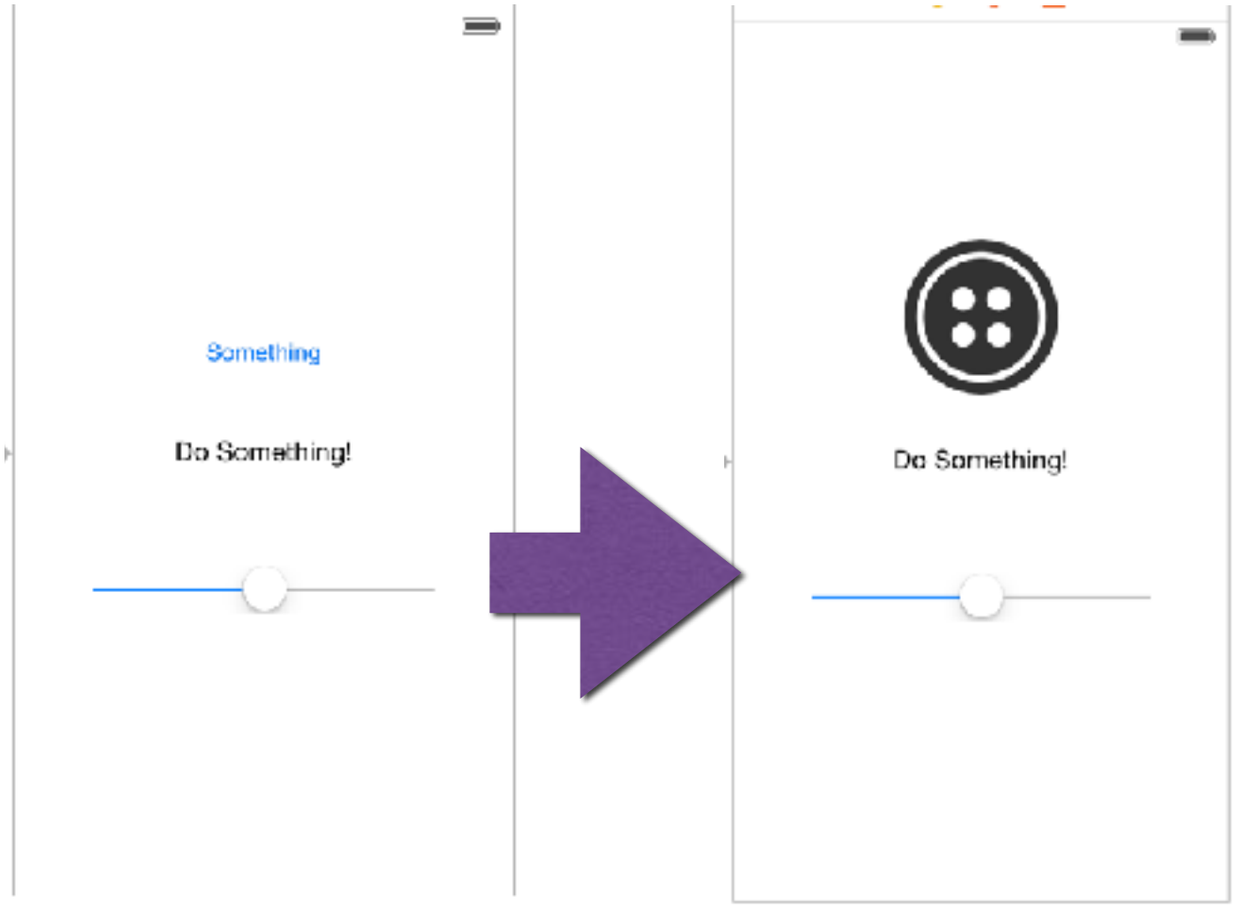
storyboard classes: the views

class: the controller

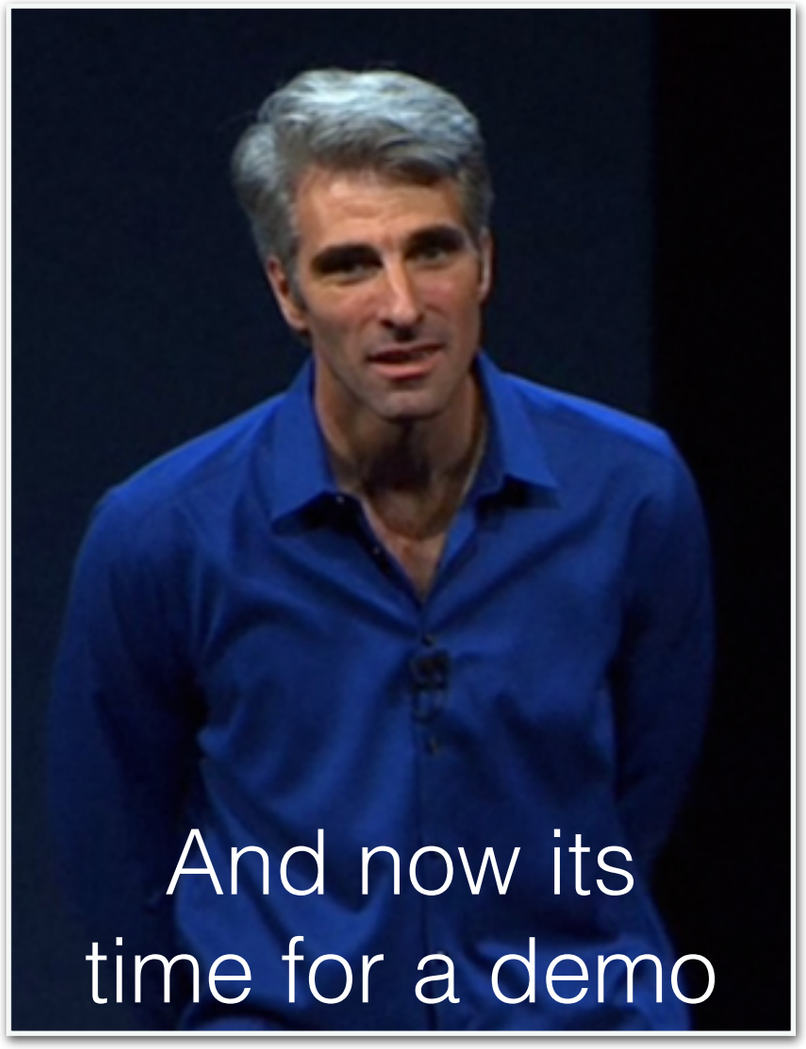
```
//
#import "ViewController.h"

@interface ViewController ()
@property (weak, nonatomic) IBOutlet UILabel *
    somethingLabel;

@end

@implementation ViewController
- (IBAction)buttonPressed:(UIButton *)sender {
    self.somethingLabel.text = @"Thanks!";
}
- (IBAction)sliderChanged:(UISlider *)sender {
    self.somethingLabel.text = [NSString
        stringWithFormat:@"Value of slider is %.
        2f",sender.value];
}

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

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!

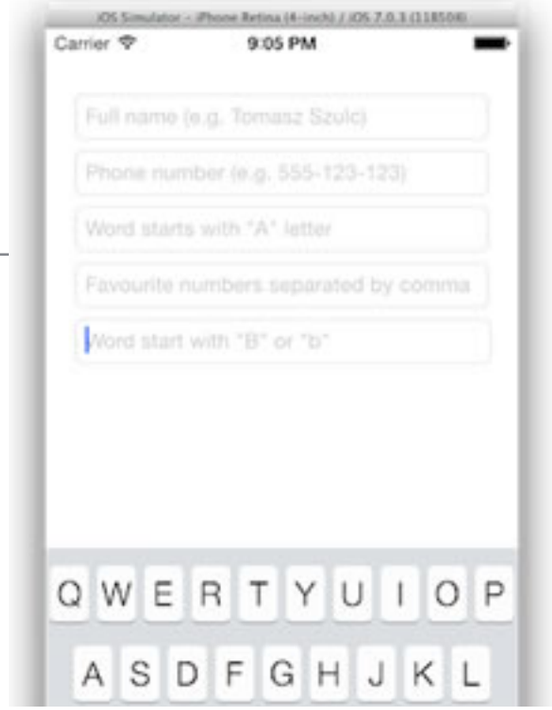


UI basics demo



text fields

- text fields are common
- but they require the use of the keyboard!
- so you need **delegate** when events happen
  - 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

- (void)viewDidLoad {
    [super viewDidLoad];
    self.nameTextField.delegate = self;
}

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

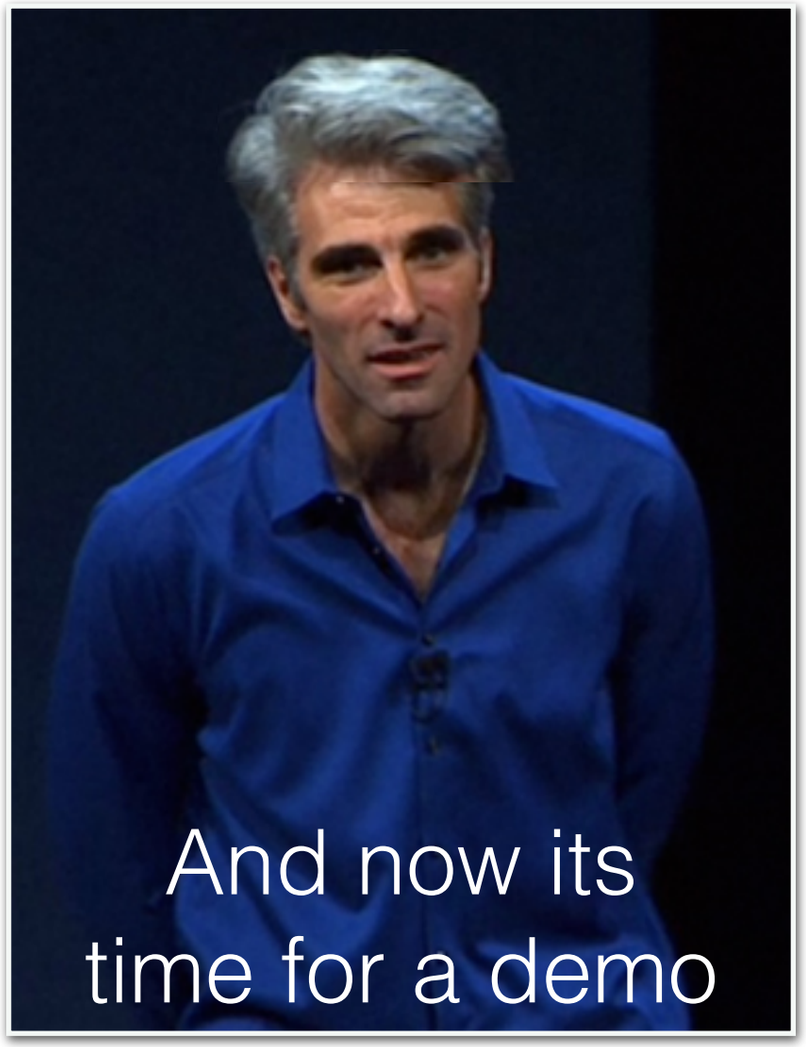
tell compiler we are delegate

return button pressed

make VC delegate

give up keyboard control

UI text field demo



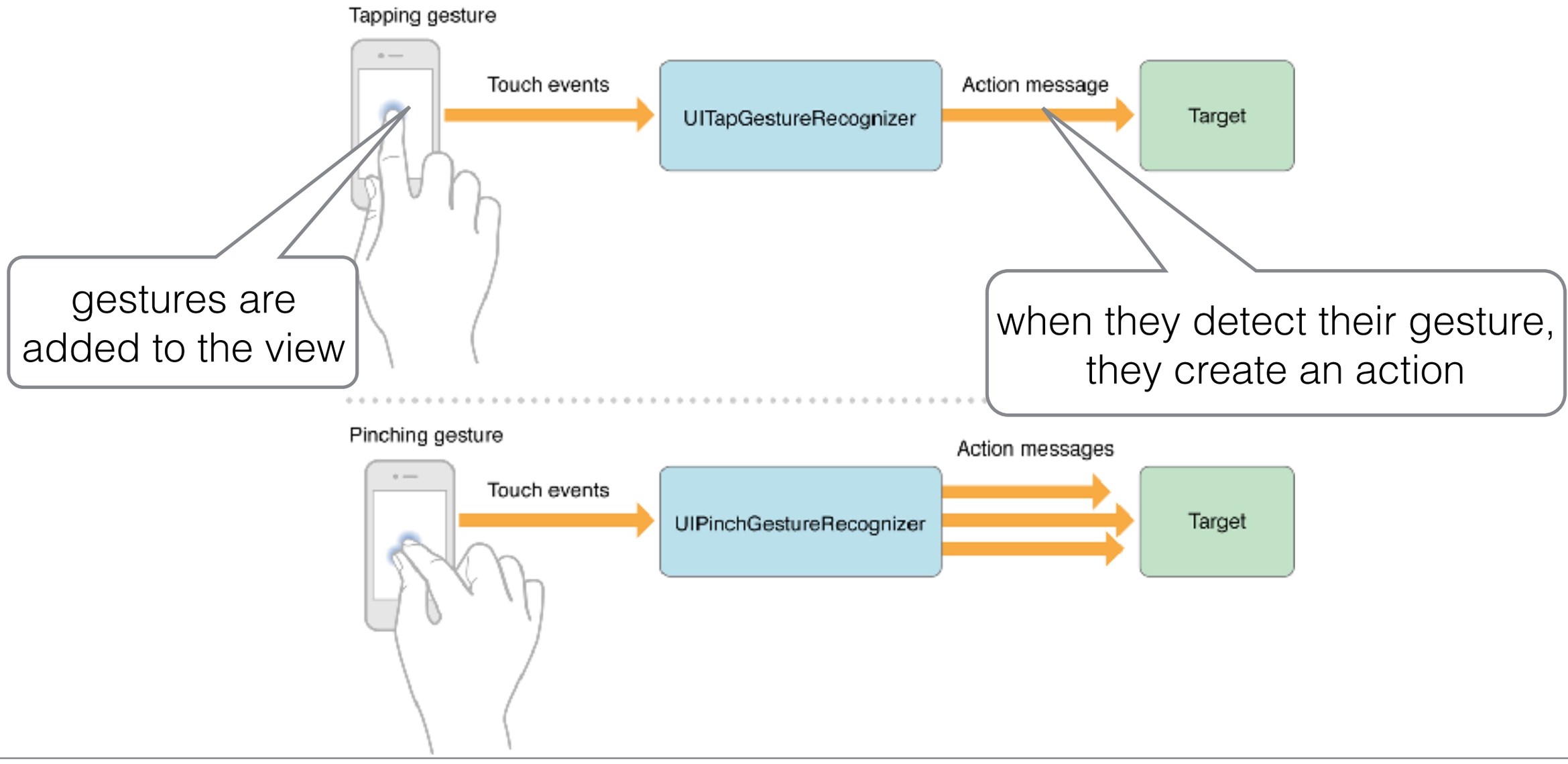
gesture recognition

- 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

gesture recognition

- need a UIGestureRecognizer
  - UITapGestureRecognizer, UIPinchGestureRecognizer, …



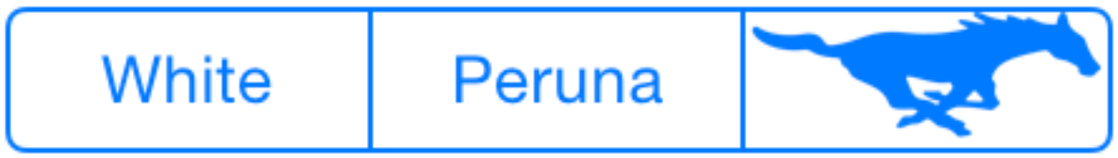
UI gesture demo

timers, segmented control

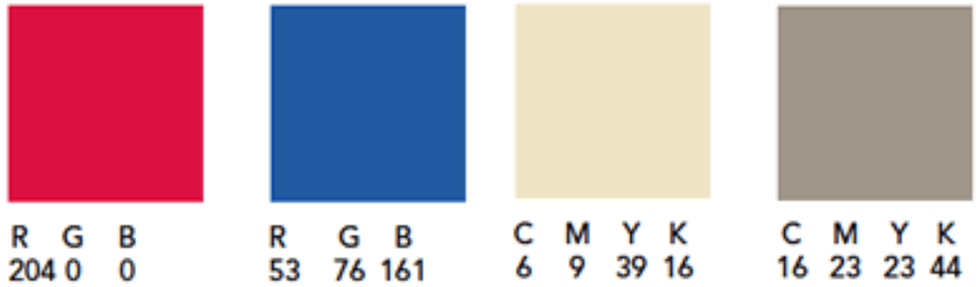
```
- (IBAction)updateFromSegmentedControl:(UISegmentedControl *)sender {

    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];
```

when should the timer be running? what modes?

pickers

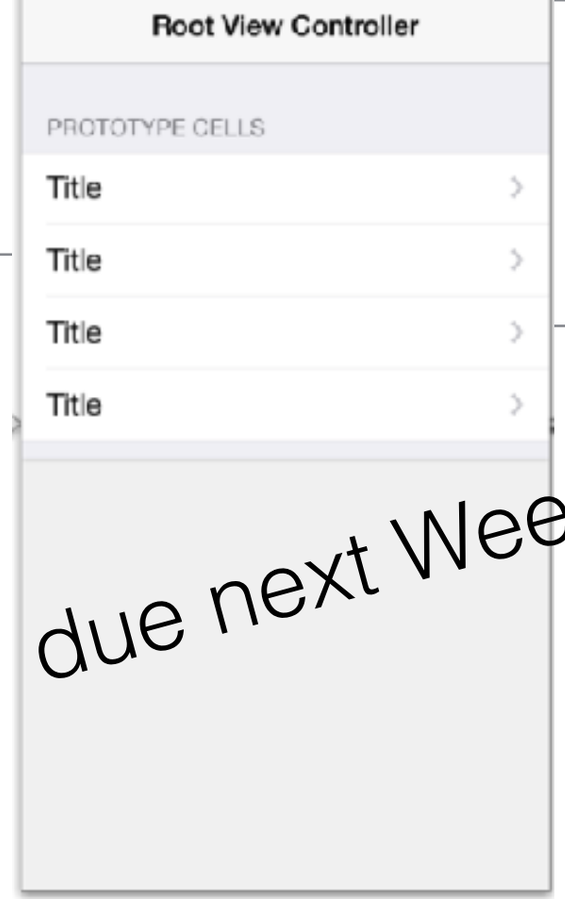
- **look at documentation**: find out how to use a picker view
- you have all the tools to do it from working with collections and the table view controllers!
- you are the data source

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



due next Week

- 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

and now…

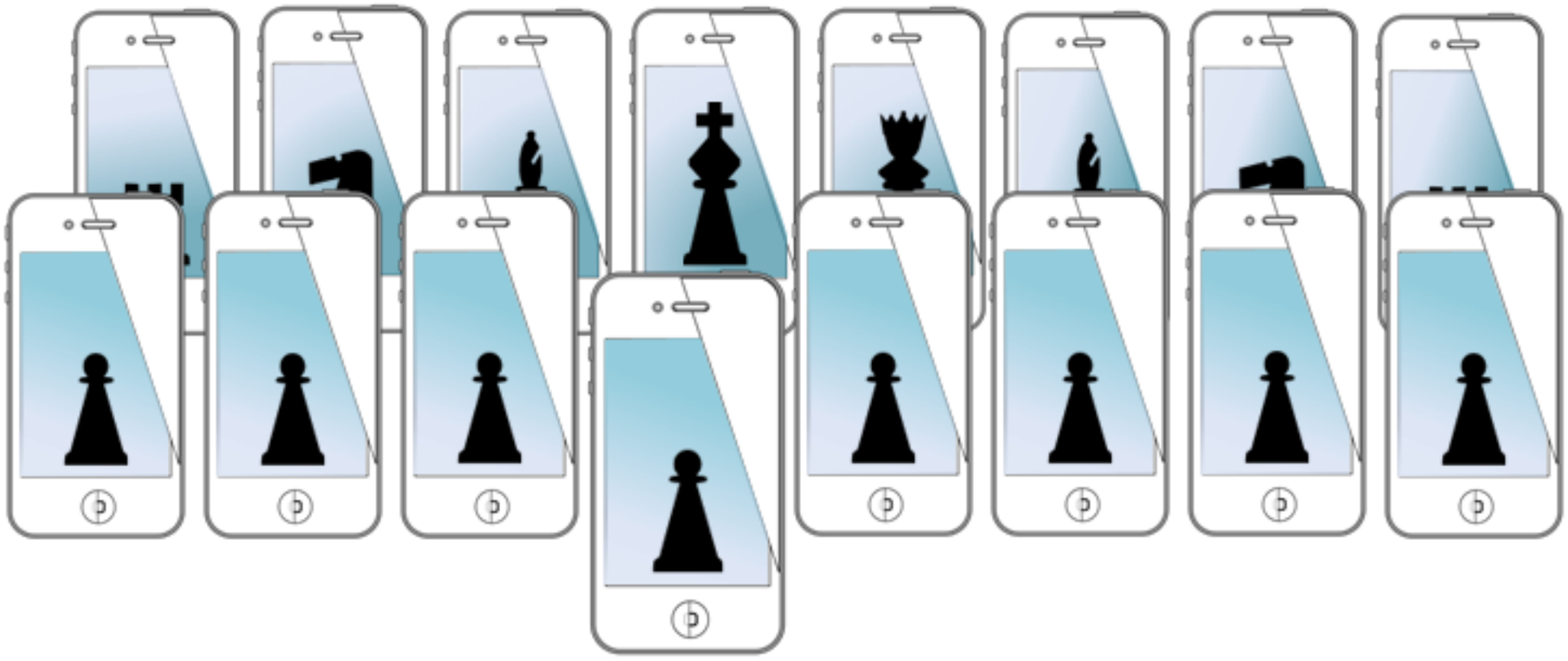
- swift!?
- from Apple, because Apple isn't afraid to force you to adopt new paradigms

https://developer.apple.com/swift/

for next week…

- mobile HCI
- concurrency though blocks
- audio sessions
- graphing audio samples

MOBILE SENSING & LEARNING



CSE5323 & 7323

Mobile Sensing & Learning

week two, lecture two: UI elements swift

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

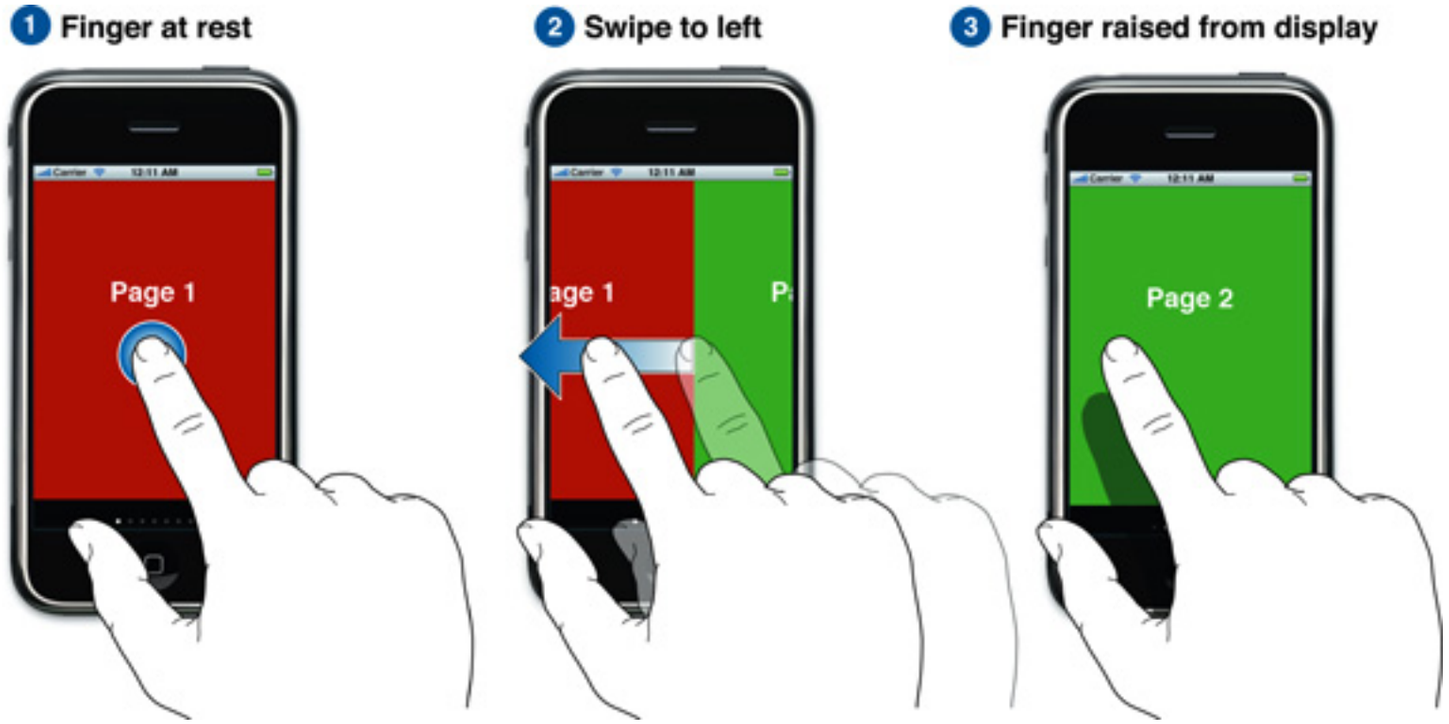
disclaimer!

page view controller

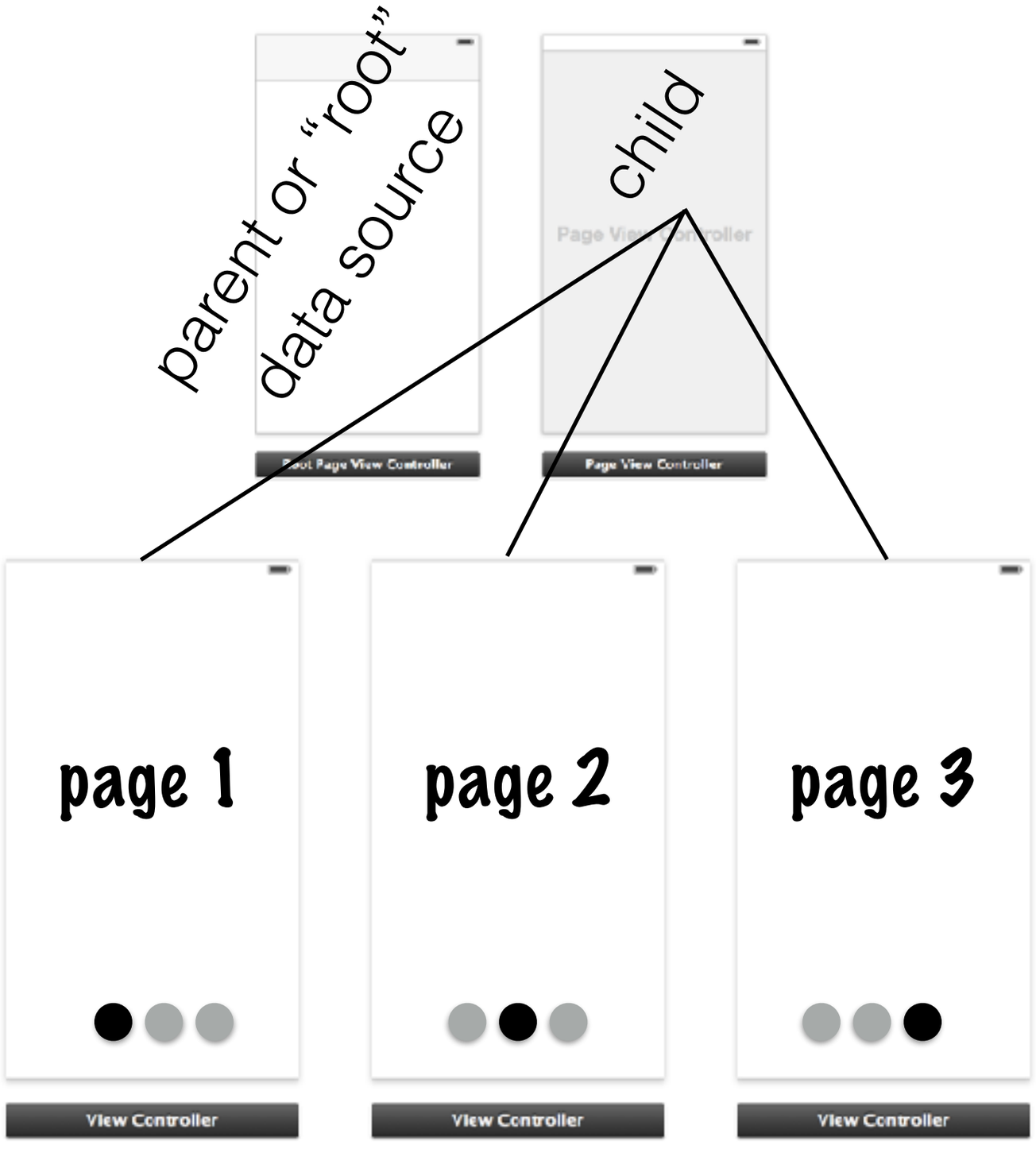
- place UIPageViewController in storyboard
- place a “root controller” for the page
  - adopt <UIPageViewControllerDataSource>
  - instantiate pageViewController
  - instantiate views to be paged

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”



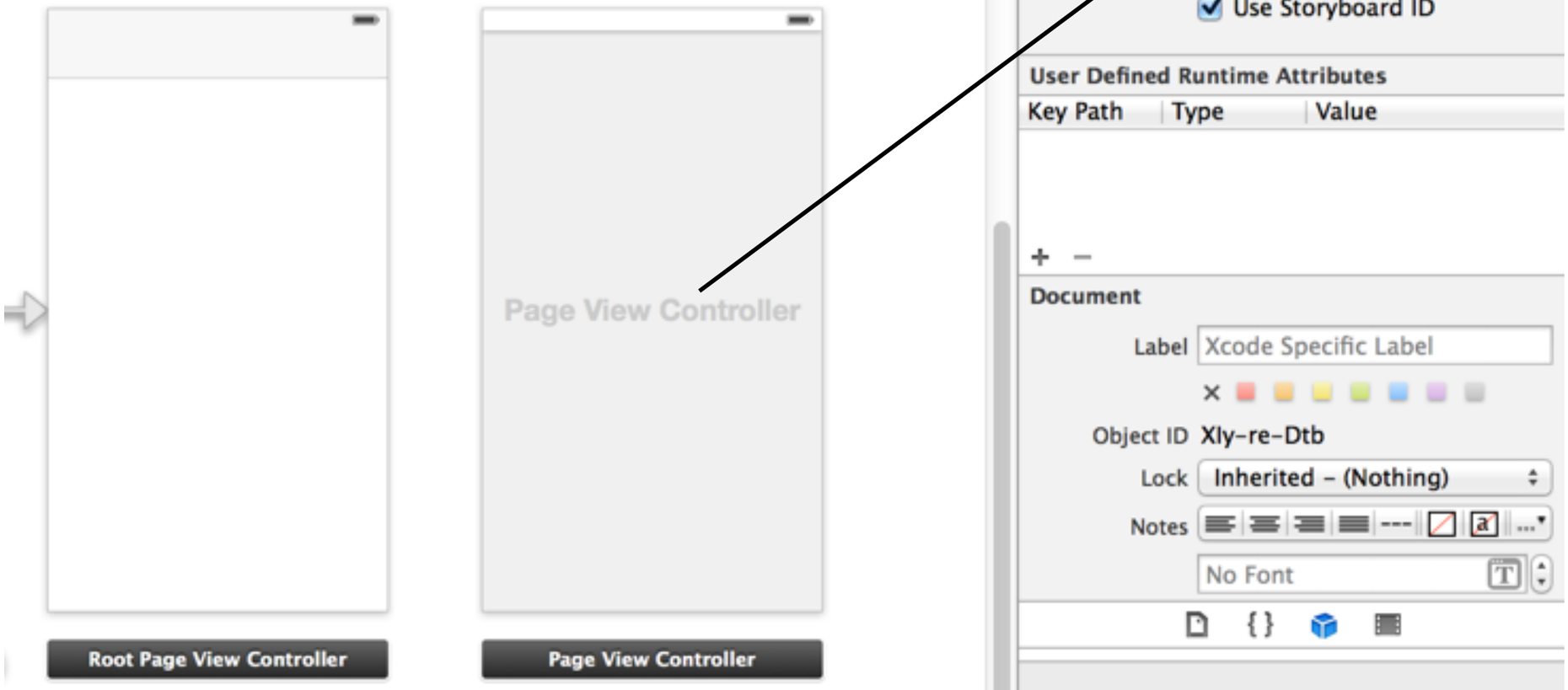
page view controller



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.)

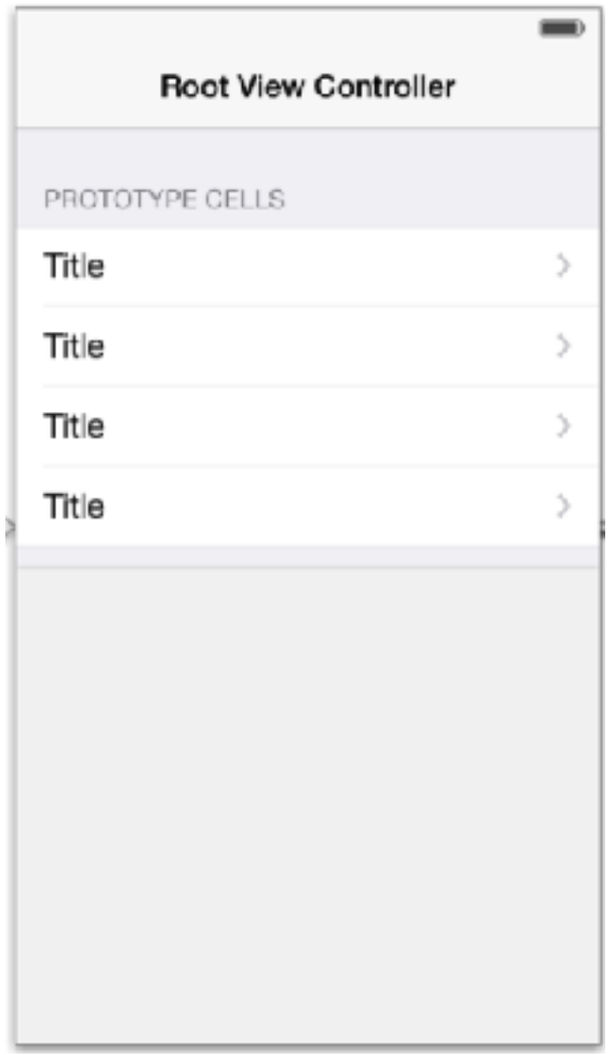
```
- (NSInteger)presentationCountForPageViewController:(UIPageViewController *)pageViewController
{
    return [self.pageContent count];
}

- (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

programatic 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:UIControlEventTouchDown];

// 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

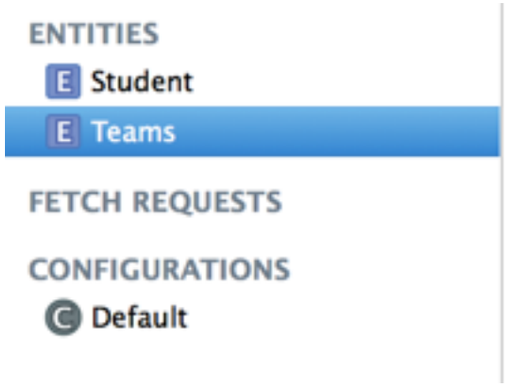
24 points between

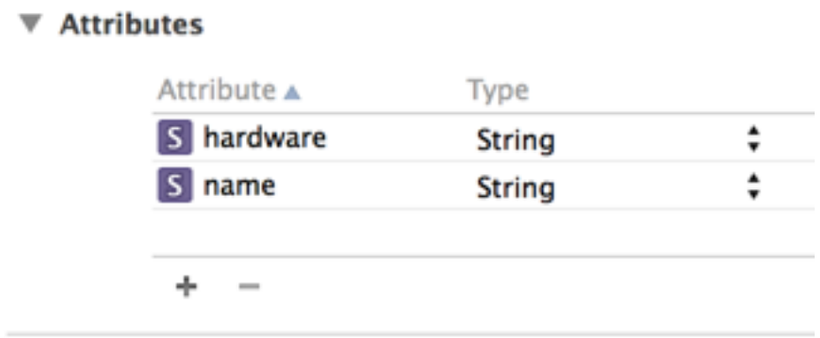
```
// metrics for use in visual constraints
NSDictionary *metrics = @{@“spacing":@10.0};

[NSLayoutConstraint constraintsWithVisualFormat:@"|-[button]-spacing-[label(==button)]-|"
                                        options:0
                                        metrics:metrics
                                          views:varBindings];


        @“V:|-[button]”
        options:NSLayoutFormatAlignAllTop | NSLayoutFormatAlignAllCenterX
```

core data databases

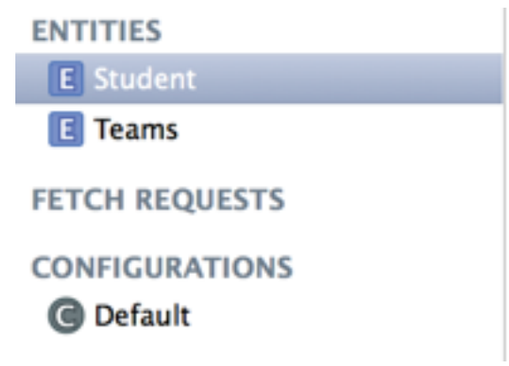
- allows access to SQLite database
- integrated deeply into Xcode and into iOS
- highly optimized
- excellent for storing persistent table data
- but usable for most anything

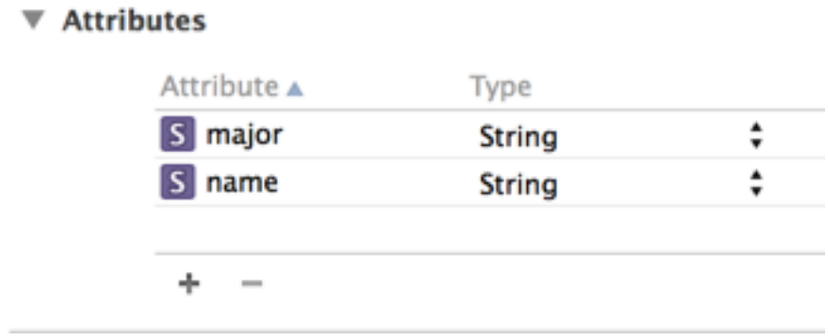
core data schema



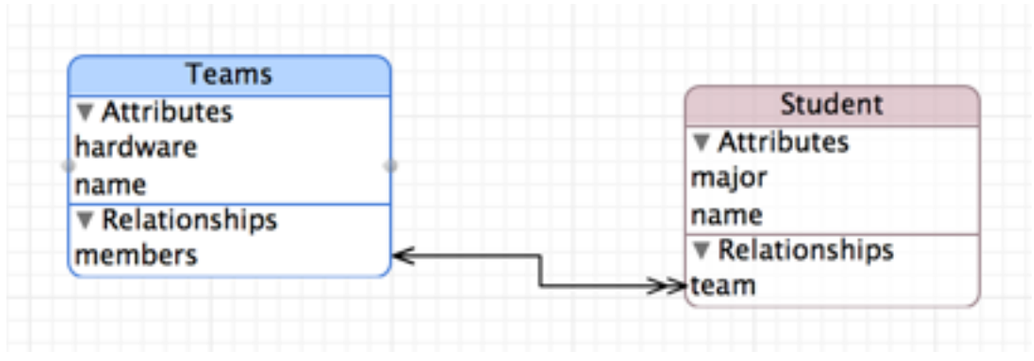



```
@interface Teams : NSManagedObject

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

@end
```






```
@interface Student : NSManagedObject

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

@end
```



core data

- schema creation
  - automatic subclassing
- NSManagedObject
- NSManagedObjectContext
- NSPersistentStore
- NSFetchRequest

create SQLite Database on phone

enable access through properties

bundle “data models”

get “context” for using data model

coordinate access to the data model

create and execute queries

core data setup

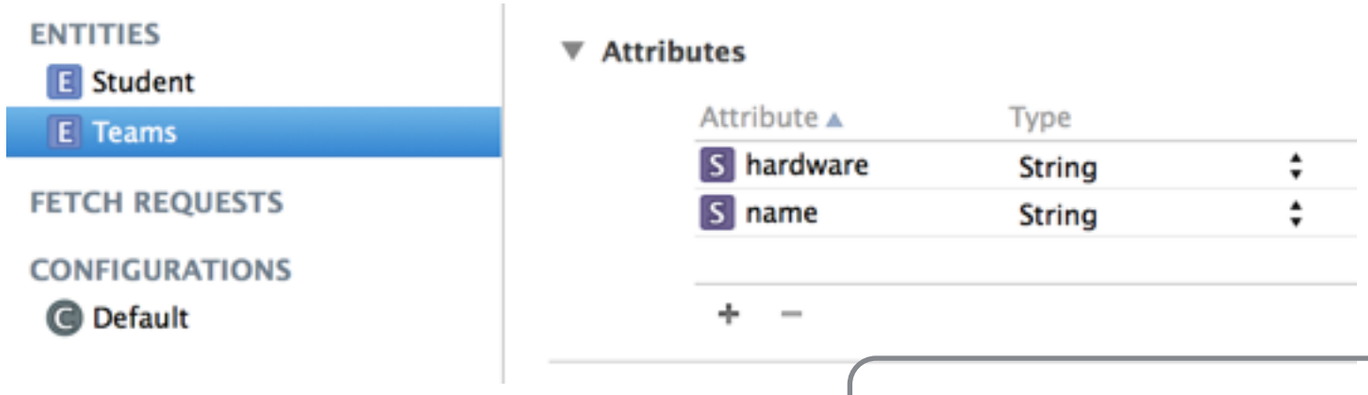
```
// Getter for managed context
- (NSManagedObjectContext *) managedObjectContext {

    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;
}


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

    …

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

    …

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

request

entity to request from

fetch

array of results, even if size=0

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

```
MCDAppDelegate* appDelegate =                    delegate;

    [[NSNotificationCenter defaultCenter] addObserver:self
                                             selector:@selector(tableDataDidChange)
                                                 name:NSManagedObjectContextDidSaveNotification
                                               object:appDelegate.managedObjectContext];
```

access notification center

do this

when this happens

from this context

lets add notifications to WhoWasInThat!