

Everything but the Kitchen Sink

Daniel Tull

I'm a contractor...

- * Half my work is helping existing projects
- * A lot of that is “firefighting”
 - * Build-up of unexplained bugs
 - * Features deemed impossible

Types of Bad Code





Errors

- * Misconfigured build environment
- * Missing dependencies
- * Poor code review
- * Lacking any form of integration (let alone continuous)



Errors

Poor project management





Warnings

- * Incompatible assignments
- * Unused variables
- * Using deprecated methods
- * Unimplemented methods
- * Interface Builder Autolayout issues
- * Undeclared selectors



Warnings

Lack of due diligence





Questionable Code

Not Coding Style!

- * Tabs not spaces
- * Opening brace on first line
- * Single line if-returns
- * Unless it looks ugly

Not Coding Style!

* I write nice looking code



Questionable Code

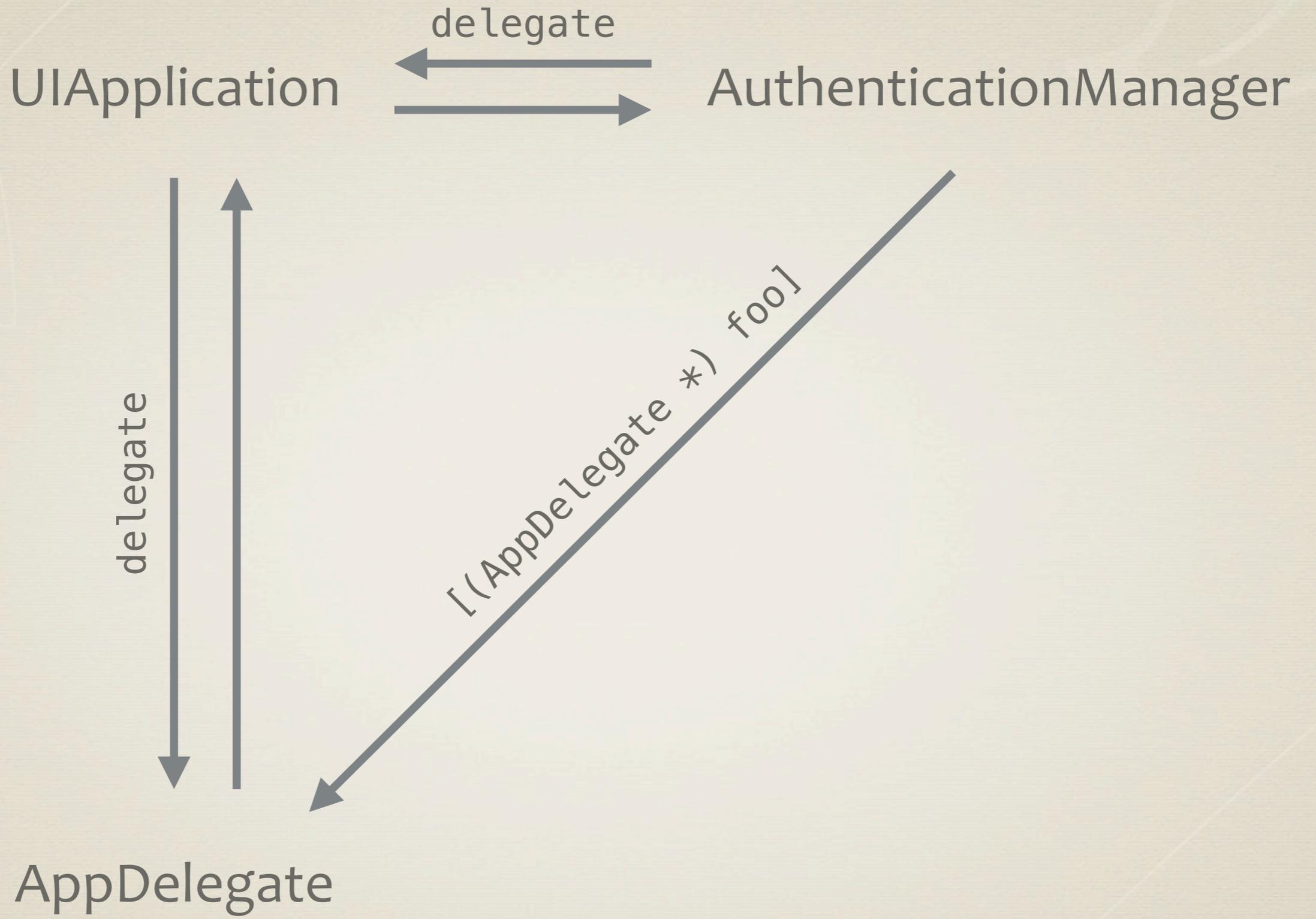
- * Tightly coupled classes
- * Random code in the app delegate
- * “Massive View Controllers”

The One and Only App Delegate

UIApplication

AuthenticationManager

AppDelegate



UIApplication

AuthenticationManager

AppDelegate

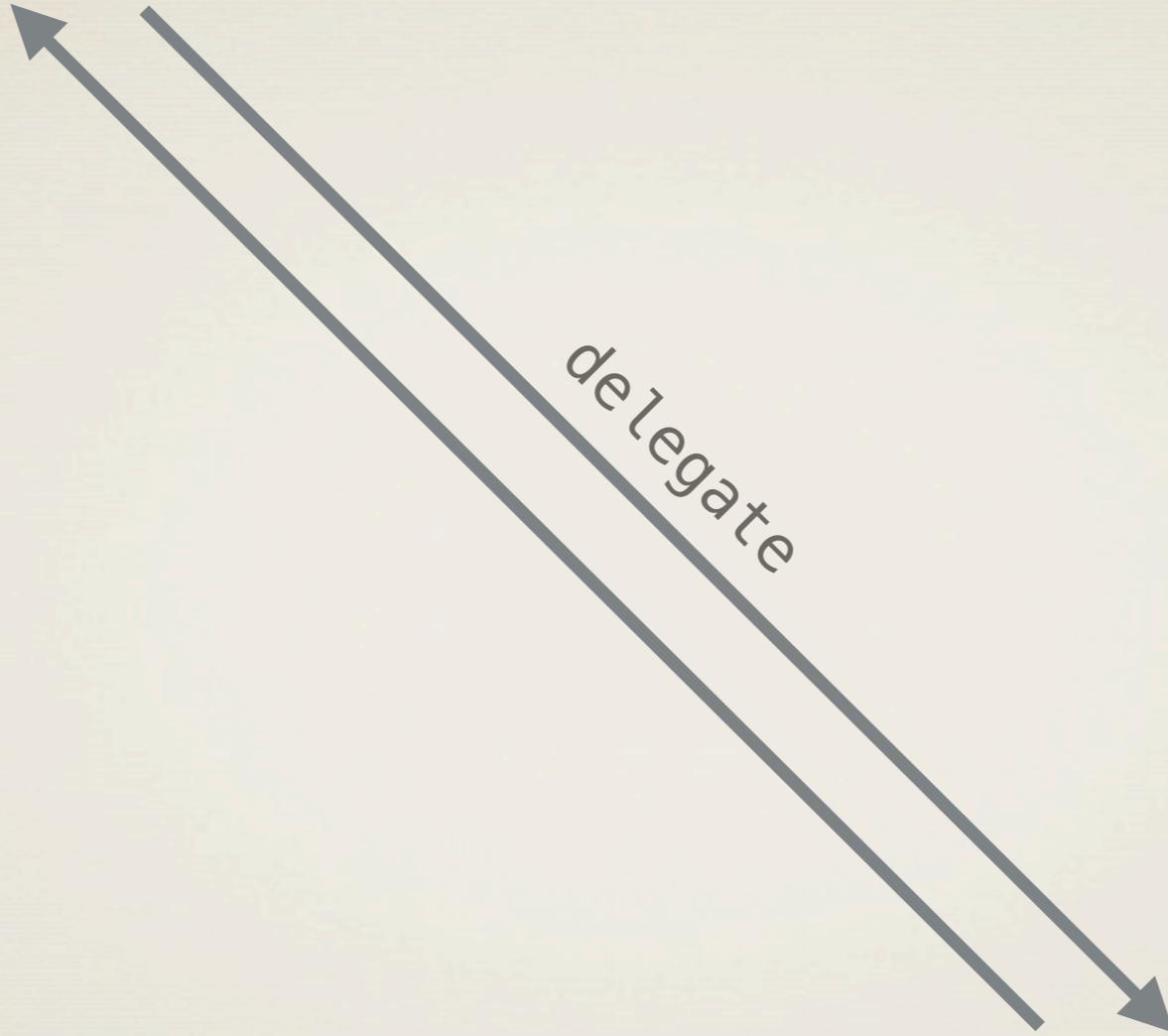
NewAppDelegate

AppDelegate

delegate



delegate



[(AppDelegate *) foo]



Solutions

- * Add the missing methods to AppDelegate

Solutions

- * Check the Application's Delegate's class
 - * `[appDelegate isKindOfClass:[OldAppDelegate class]]`
- * On deleting OldAppDelegate will cause an error

Massive View Controllers



Colin Campbell

@Colin_Campbell



Follow

iOS architecture, where MVC stands for
Massive View Controller

Reply Retweet Favorite More

209

RETWEETS

82

FAVORITES



1:27 AM - 21 Jan 13

```
//
// LDGEventsViewController.m
// LiDG
//
// Created by Daniel Tull on 23.08.2013.
// Copyright (c) 2013 Daniel Tull. All rights reserved.
//

#import "LDGEventsViewController.h"

@interface LDGEventsViewController ()

@end

@implementation LDGEventsViewController

- (id)initWithStyle:(UITableViewStyle)style
{
    self = [super initWithStyle:style];
    if (self) {
        // Custom initialization
    }
    return self;
}

- (void)viewDidLoad
{
    [super viewDidLoad];

    // Uncomment the following line to preserve selection between presentations.
    // self.clearsSelectionOnViewWillAppear = NO;

    // Uncomment the following line to display an Edit button in the navigation bar for this view
    controller.
    // self.navigationItem.rightBarButtonItem = self.editButtonItem;
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

#pragma mark - Table view data source

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

```

- (void)tableView:(UITableView *)tableView commitEditingStyle:(UITableViewCellEditingStyle)editingStyle
forRowAtIndexPath:(NSIndexPath *)indexPath
{
    if (editingStyle == UITableViewCellEditingStyleDelete) {
        // Delete the row from the data source
        [tableView deleteRowsAtIndexPaths:@[indexPath] withRowAnimation:UITableViewRowAnimationFade];
    }
    else if (editingStyle == UITableViewCellEditingStyleInsert) {
        // Create a new instance of the appropriate class, insert it into the array, and add a new row to
the table view
    }
}
*/

/*
// Override to support rearranging the table view.
- (void)tableView:(UITableView *)tableView moveRowAtIndexPath:(NSIndexPath *)fromIndexPath toIndexPath:
(NSIndexPath *)toIndexPath
{
}
*/

/*
// Override to support conditional rearranging of the table view.
- (BOOL)tableView:(UITableView *)tableView canMoveRowAtIndexPath:(NSIndexPath *)indexPath
{
    // Return NO if you do not want the item to be re-orderable.
    return YES;
}
*/

#pragma mark - Table view delegate

- (void)tableView:(UITableView *)tableView didSelectRowAtIndexPath:(NSIndexPath *)indexPath
{
    // Navigation logic may go here. Create and push another view controller.
    /*
    <#DetailViewController#> *detailViewController = [[<#DetailViewController#> alloc]
initWithNibName:@"<#Nib name#>" bundle:nil];
    // ...
    // Pass the selected object to the new view controller.
    [self.navigationController pushViewController:detailViewController animated:YES];
    */
}

```

@end

Apple Doesn't Help!

```
//  
// LDGEventsViewController.m  
// LiDG  
//  
// Created by Daniel Tull on 23.08.2013.  
// Copyright (c) 2013 Daniel Tull. All rights reserved.  
//
```

```
#import "LDGEventsViewController.h"
```

```
@interface LDGEventsViewController ()
```

```
@end
```

```
@implementation LDGEventsViewController
```

```
@end
```

Create Controller Classes

Using a DataSource class

- * Implement the UITableViewDataSource methods
- * Update the table view with new data
- * Use properties to change behaviour

Carrier 

6:05 PM



About

Previous

Upcoming

LiDG 55

Wednesday, September 11, 2013



Carrier 

6:05 PM



About

Previous

Upcoming

LiDG 54

Wednesday, August 7, 2013



LiDG 53

Wednesday, July 3, 2013



Using a DataSource class

```
- (void)setupDataSource {  
  
    NSManagedObjectContext *context = self.dataManager.managedObjectContext;  
    NSFetchRequest *fetchRequest = [[NSFetchRequest alloc] initWithEntityName:[LDGEvent entityName]];  
  
    if (self.segmentedControl.selectedSegmentIndex == 0) {  
        fetchRequest.predicate = [NSPredicate predicateWithFormat:@"%K < %@",  
                                LDGEventAttributes.date, [NSDate new]];  
        fetchRequest.sortDescriptors = @[[[NSSortDescriptor alloc] initWithKey:LDGEventAttributes.date  
                                         ascending:NO]];  
    } else {  
        fetchRequest.predicate = [NSPredicate predicateWithFormat:@"%K > %@",  
                                LDGEventAttributes.date, [NSDate new]];  
        fetchRequest.sortDescriptors = @[[[NSSortDescriptor alloc] initWithKey:LDGEventAttributes.date  
                                         ascending:YES]];  
    }  
  
    self.dataSource = [[DCTFetchedResultsTableViewDataSource alloc] initWithManagedObjectContext:context  
                                                                fetchRequest:fetchRequest];  
    self.dataSource.cellReuseIdentifierHandler = ^(NSIndexPath *indexPath, id object) {  
        return @"event";  
    };  
  
    self.tableView.dataSource = self.dataSource;  
    self.dataSource.tableView = self.tableView;  
    [self.tableView reloadData];  
  
}
```

Using a DataSource class

```
- (void)viewDidLoad {  
    [super viewDidLoad];  
    [self setupDataSource];  
    [self.dataManager fetchEventsWithCompletion:nil];  
    [self.dataManager fetchPeopleWithCompletion:nil];  
}  
  
- (IBAction)segmentedControlValueChanged:(id)sender {  
    [self setupDataSource];  
}
```

Using a DataSource class

```
- (void)tableView:(UITableView *)tableView
willDisplayCell:(UITableViewCell *)cell
forRowAtIndexPath:(NSIndexPath *)indexPath {

    LDGEvent *event = [self.dataSource objectAtIndex:indexPath];
    cell.textLabel.text = event.name;
    cell.detailTextLabel.text = [[[self class] dateFormatter] stringFromDate:event.date];
}

- (void)prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender {

    id viewController = segue.destinationViewController;

    if ([viewController isKindOfClass:[LDGEventViewController class]]) {
        LDGEventViewController *eventViewController = viewController;
        NSIndexPath *indexPath = [self.tableView indexPathForSelectedRow];
        eventViewController.event = [self.dataSource objectAtIndex:indexPath];
        eventViewController.dataManager = self.dataManager;
    }
}
```

What else to separate?

- * Anything that you feel yourself re-writing!
- * View Controllers should glue logic bits together

Use IBOutlet

- * Using Interface Builder will save time
- * Have objects with IBOutlet properties
- * Instantiate them in the nib/storyboard
- * Make a strong property for them in the view controller
- * I've rarely had to reference them beyond that

Use IBOutlet

The image displays the Xcode interface for a registration form. On the left, the 'Registration View Controller Scene' sidebar shows a hierarchy: Registration View Controller > Table View > Table View Section > View > Green Button - Register. Below this are various UI components like Navigation Item, First Responder, Exit, Picker, Title View, and Text Field Validator.

The central canvas shows a registration form with the following fields and controls:

- Username
- Email
- Password
- Confirm
- First Name
- Last Name
- Birthday
- Accept terms (OFF)
- Newsletter (OFF)
- Register button

On the right, the 'Outlets' panel is configured as follows:

- enabledObject:** Green Button - Register
- requiredTextFields:** Username Text Field, Email Text Field, Password Text Field, Confirm Text Field
- textFields:** Username Text Field, Email Text Field, Password Text Field, Confirm Text Field, First Name Text Field, Last Name Text Field, Birthday Text Field
- Referencing Outlets:** textFieldValidator (Referencing Registration View Controller)
- Referencing Outlet Collections:** New Referencing Outlet Collection

Use IBOutlet

The image shows a screenshot of the Xcode IDE. On the left is the 'Registration View Controller Scene' storyboard, which contains a registration form with the following fields and controls:

- Username
- Email
- Password
- Confirm
- First Name
- Last Name
- Birthday
- Accept terms (toggle switch, currently OFF)
- Newsletter (toggle switch, currently OFF)
- Register button

On the right is the 'Outlets' panel, which is highlighted with an orange border. It shows the following configuration:

- enabledObject:** Green Button - Register
- requiredTextFields:** A collection containing Username Text Field, Email Text Field, Password Text Field, and Confirm Text Field.
- textFields:** A collection containing Username Text Field, Email Text Field, Password Text Field, Confirm Text Field, First Name Text Field, Last Name Text Field, and Birthday Text Field.
- Referencing Outlets:** textFieldValidator is connected to Registration View Controller.
- Referencing Outlet Collections:** A new referencing outlet collection is available.

Use IBOutlet

The image shows the Xcode interface for a registration form. On the left, the 'Registration View Controller Scene' sidebar lists various UI components, with 'Text Field Validator' highlighted. An orange arrow points from this component to the 'Register' button in the form. The form itself contains fields for Username, Email, Password, Confirm, First Name, Last Name, Birthday, and two toggle switches for 'Accept terms' and 'Newsletter'. On the right, the 'Outlets' panel shows connections for the 'Green Button - Register' to the 'enabledObject' outlet, and for the 'Text Field Validator' to the 'textFieldValidator' outlet. The 'textFieldValidator' outlet is also connected to a collection of text fields.

Registration View Controller Scene

- Registration View Controller
 - Table View
 - Table View Section
 - View
 - Green Button - Register
 - Navigation Item
 - First Responder
 - Exit
 - Picker
 - Title View
 - Text Field Validator**

Outlets

- enabledObject
 - Green Button - Register
- Outlet Collections
 - requiredTextFields
 - Username Text Field
 - Email Text Field
 - Password Text Field
 - Confirm Text Field
 - textFieldValidator
 - Username Text Field
 - Email Text Field
 - Password Text Field
 - Confirm Text Field
 - First Name Text Field
 - Last Name Text Field
 - Birthday Text Field
- Referencing Outlets
 - textFieldValidator
 - Registration View Controller
 - New Referencing Outlet
- Referencing Outlet Collections
 - New Referencing Outlet Collection

Use IBOutlet

The image shows a screenshot of the Xcode IDE. On the left is the 'Registration View Controller Scene' storyboard, which contains a registration form with the following fields: Username, Email, Password, Confirm, First Name, Last Name, Birthday, Accept terms (with an OFF toggle), and Newsletter (with an OFF toggle). At the bottom of the form is a 'Register' button. On the right is the 'Outlets' panel, which lists the following configurations:

- Outlets:** enabledObject (connected to Green Button - Register)
- Outlet Collections:**
 - requiredTextFields (connected to Username Text Field, Email Text Field, Password Text Field, Confirm Text Field)
 - textFields (connected to Username Text Field, Email Text Field, Password Text Field, Confirm Text Field, First Name Text Field, Last Name Text Field, Birthday Text Field)
- Referencing Outlets:** textFieldValidator (connected to Registration View Controller)
- Referencing Outlet Collections:** New Referencing Outlet Collection

An orange arrow points from the 'Text Field Validator' in the left-hand menu to the 'textFieldValidator' outlet in the right-hand panel. Another orange circle highlights the 'textFieldValidator' outlet and its connection to the 'Registration View Controller'.

Example controller classes

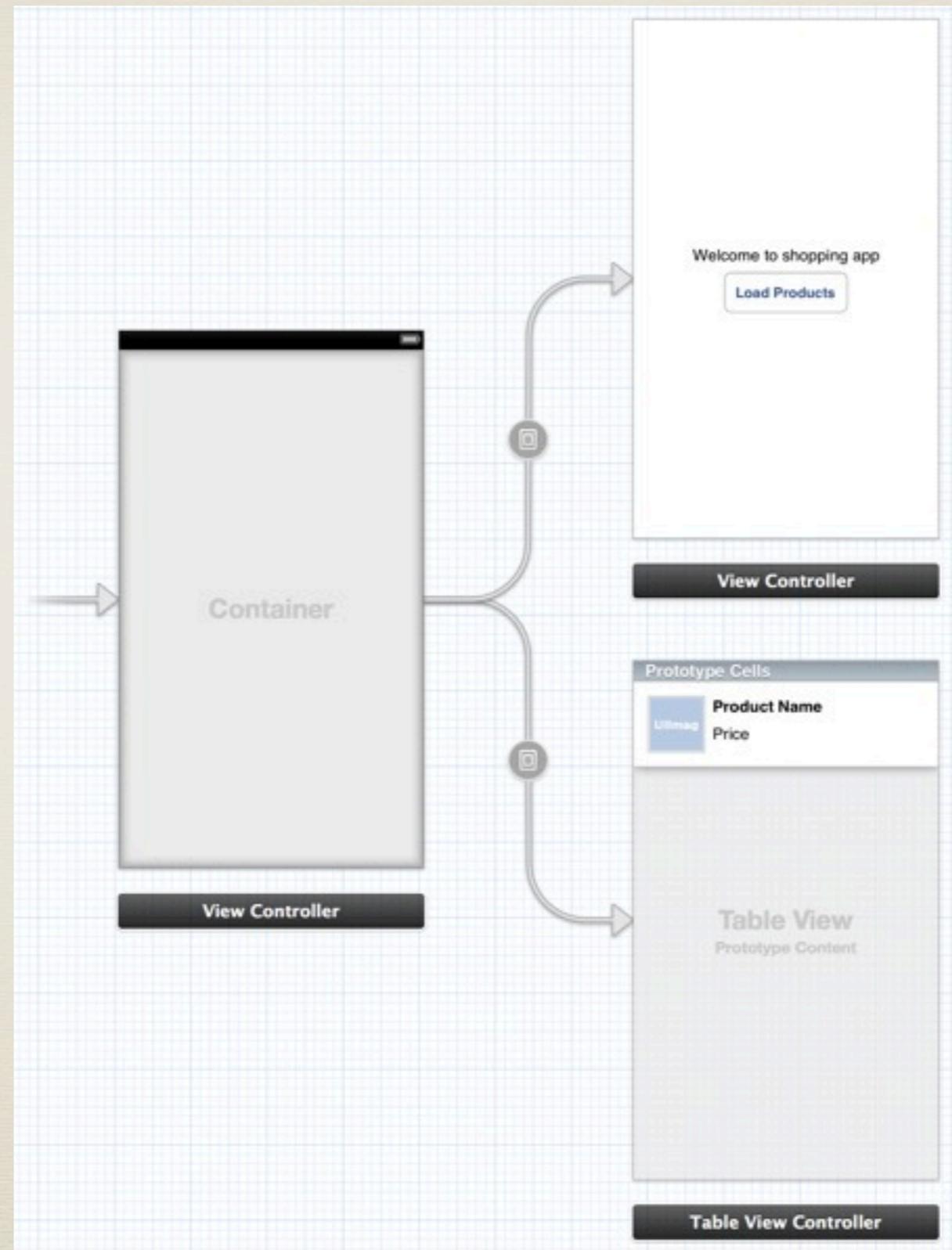
- * DCTTableViewDataSources
- * DSLFetchedResultsControllerDelegate
- * DCTTextFieldValidator

Container View Controllers

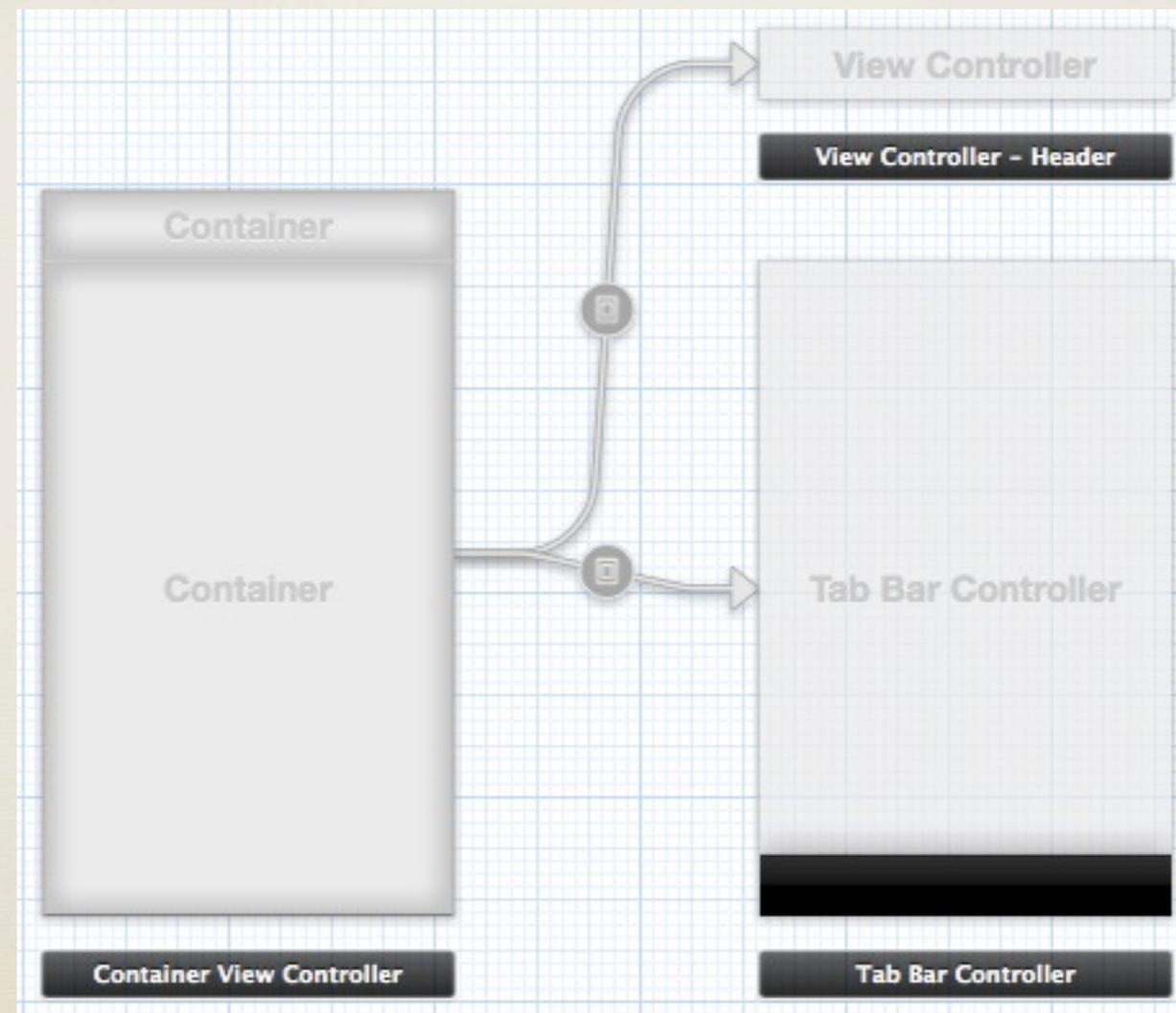
Use container view controllers

- * You can split the UI between view controllers
- * Allows flexibility with iPad/iPhone layouts

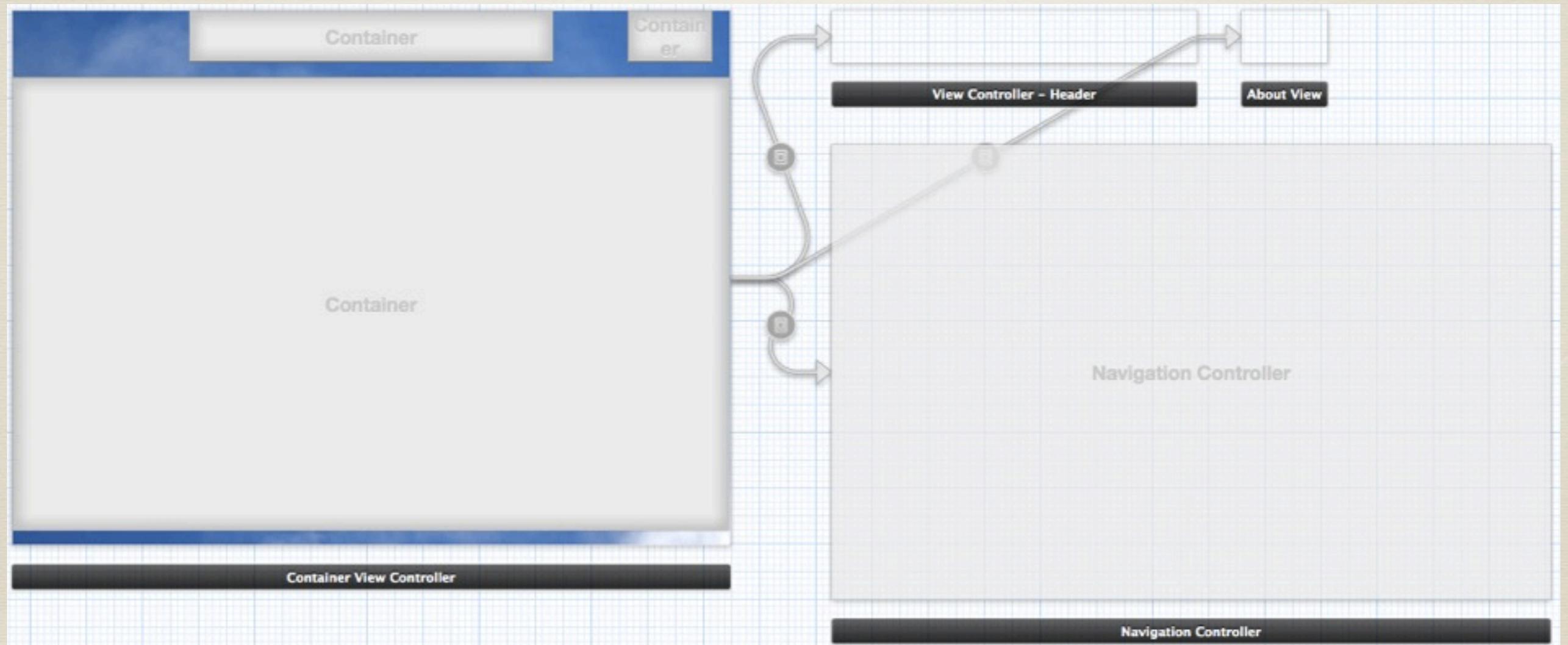
Toggling View Controllers



Containers for Layout



Containers for Layout



Container View Controller

```
@interface ContainerViewController : UIViewController
@property (nonatomic, copy) void (^viewDidAppearHandler)(BOOL animated);
@property (nonatomic, copy) void (^prepareForSegueHandler)(UIStoryboardSegue *segue, id sender);
@end
```

```
@implementation ContainerViewController
```

```
- (void)viewDidAppear:(BOOL)animated {
    [super viewDidAppear:animated];
    if (self.viewDidAppearHandler)
        self.viewDidAppearHandler(animated);
}

- (void)prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender {
    if (self.prepareForSegueHandler)
        self.prepareForSegueHandler(segue, sender);
}
```

```
@end
```

Container View Controller

```
- (void)prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender {
    id viewController = segue.destinationViewController;
    if ([viewController isKindOfClass:[ContainerViewController class]]) {
        ContainerViewController *containerViewController = viewController;
        containerViewController.prepareForSegueHandler = ^(UIStoryboardSegue *segue, id sender) {
            if ([viewController isKindOfClass:[HeaderViewController class]]) {
                HeaderViewController *headerViewController = viewController;
                headerViewController.managedObjectContext = self.managedObjectContext;
            } else if ([viewController isKindOfClass:[DashboardViewController class]]) {
                DashboardViewController *dashboardViewController = viewController;
                dashboardViewController.sport = self.sport;
            } else if ([viewController isKindOfClass:[AboutViewController class]]) {
                AboutViewController *aboutViewController = viewController;
                aboutViewController.account = self.account;
            }
        };
    }
}
```

Summary



Fix errors

Summary



Fix errors



Remove warnings

Summary



Fix errors



Remove warnings



Write as little code as possible

Daniel Tull

@danielctull

danieltull.co.uk