Design Patterns for Mobile Apps

@casademora

@saulmora

MAGICALPANDA

Cocoa Design Patterns

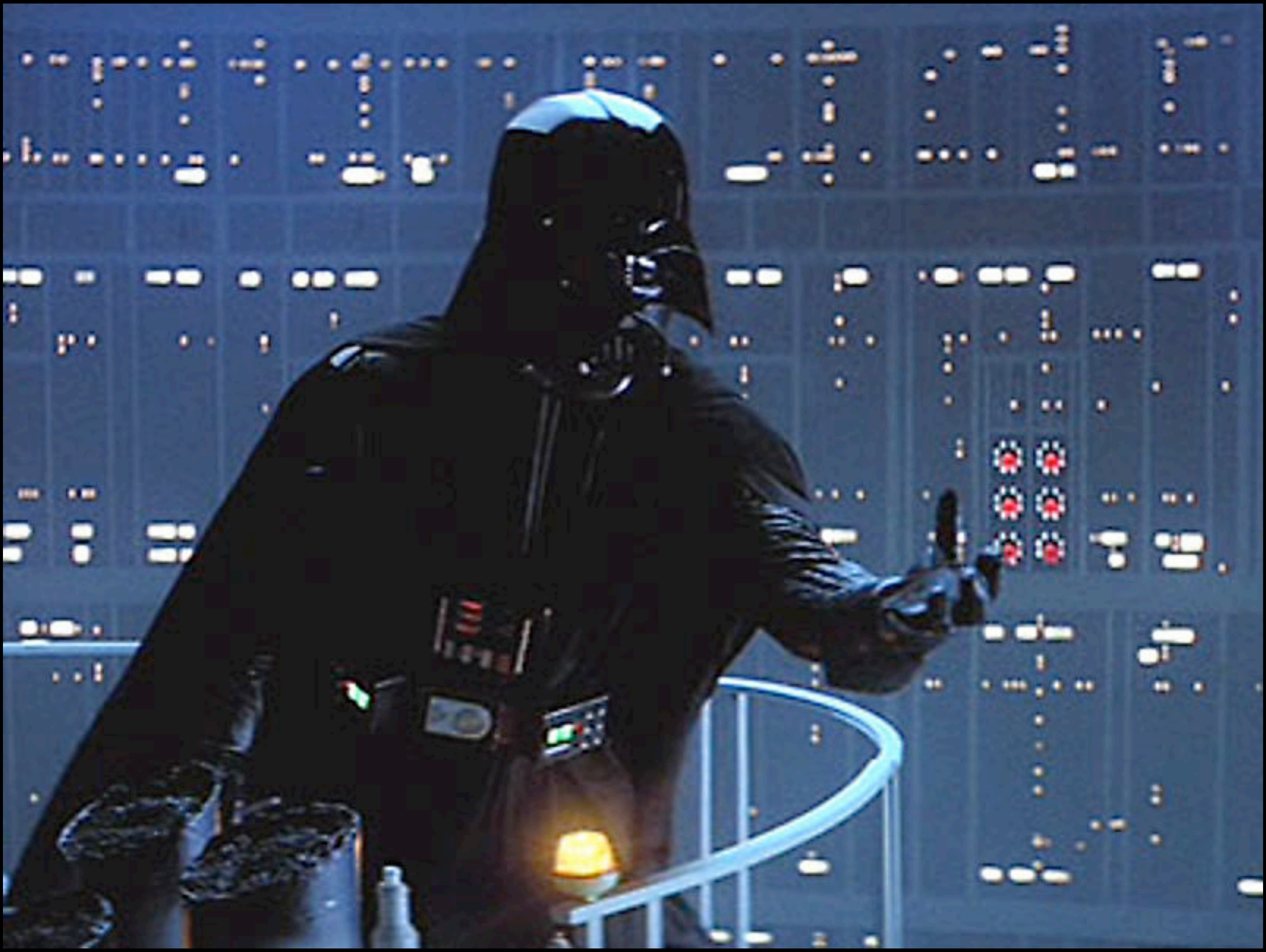
NSBrief

A long time ago, in a career far, far away…

I want to tell you a short story about a young developer

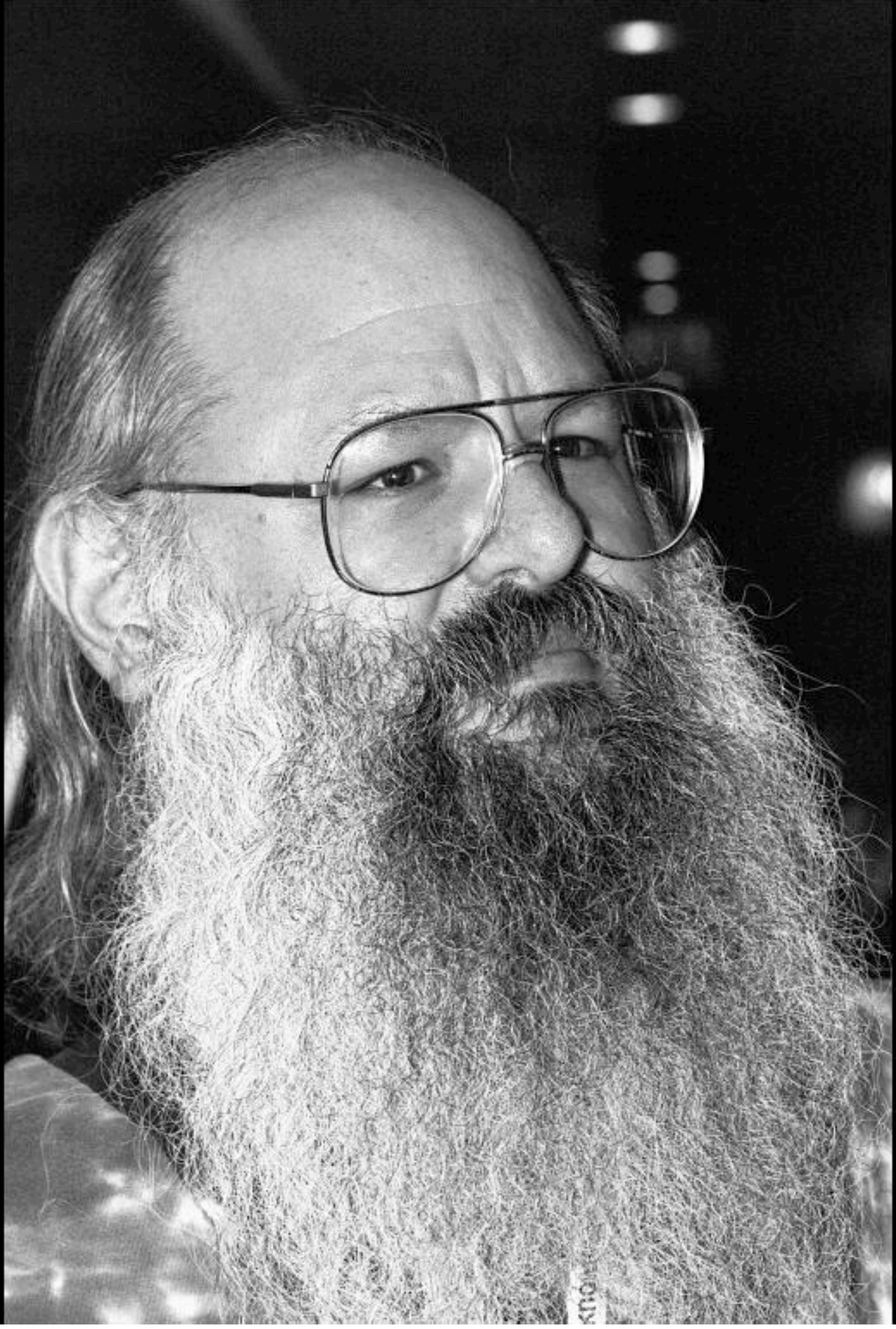
Wednesday, June 19, 13

When I was a young padawan developer



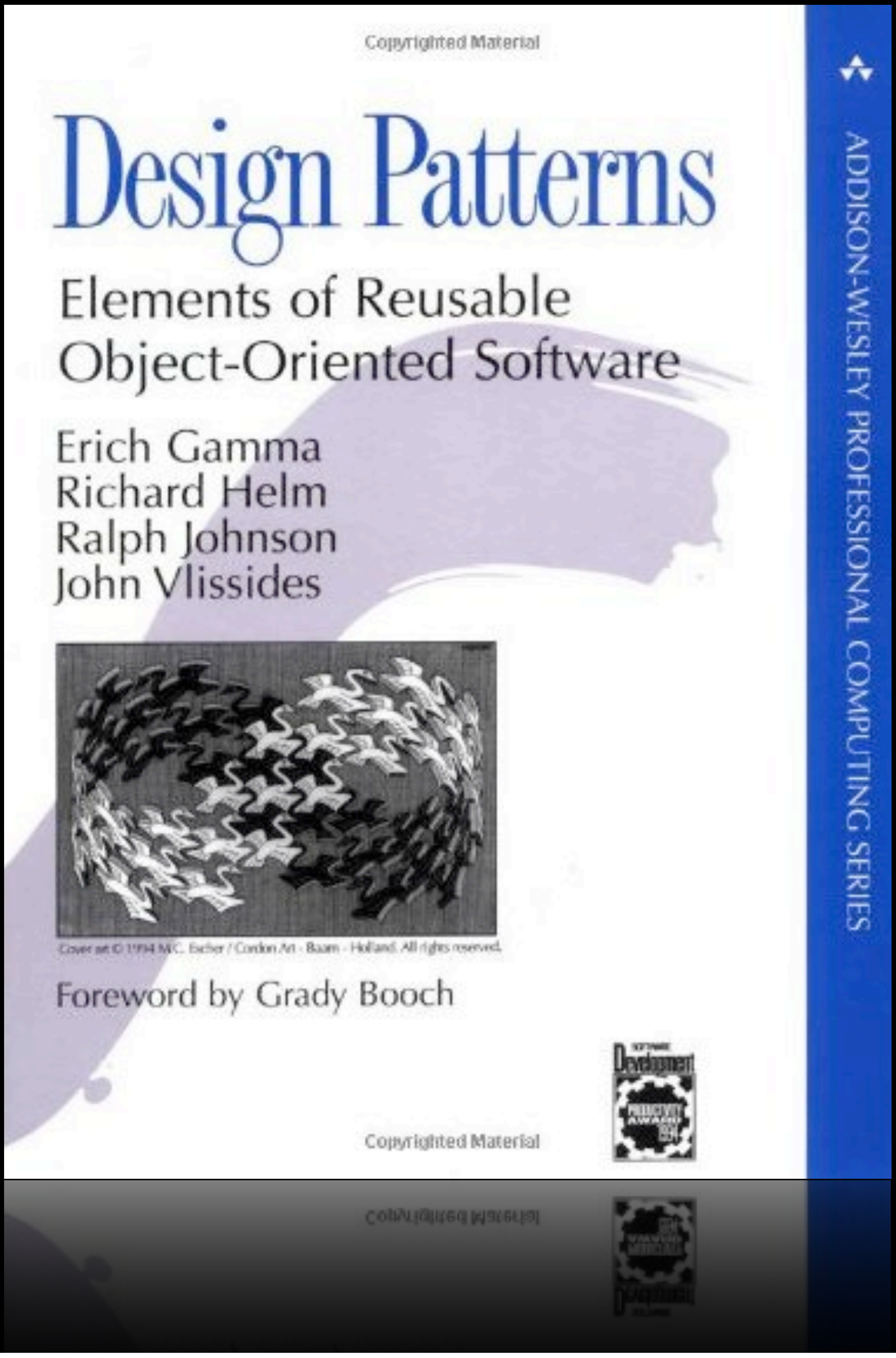
Wednesday, June 19, 13

I had a job at the empire....



Wednesday, June 19, 13

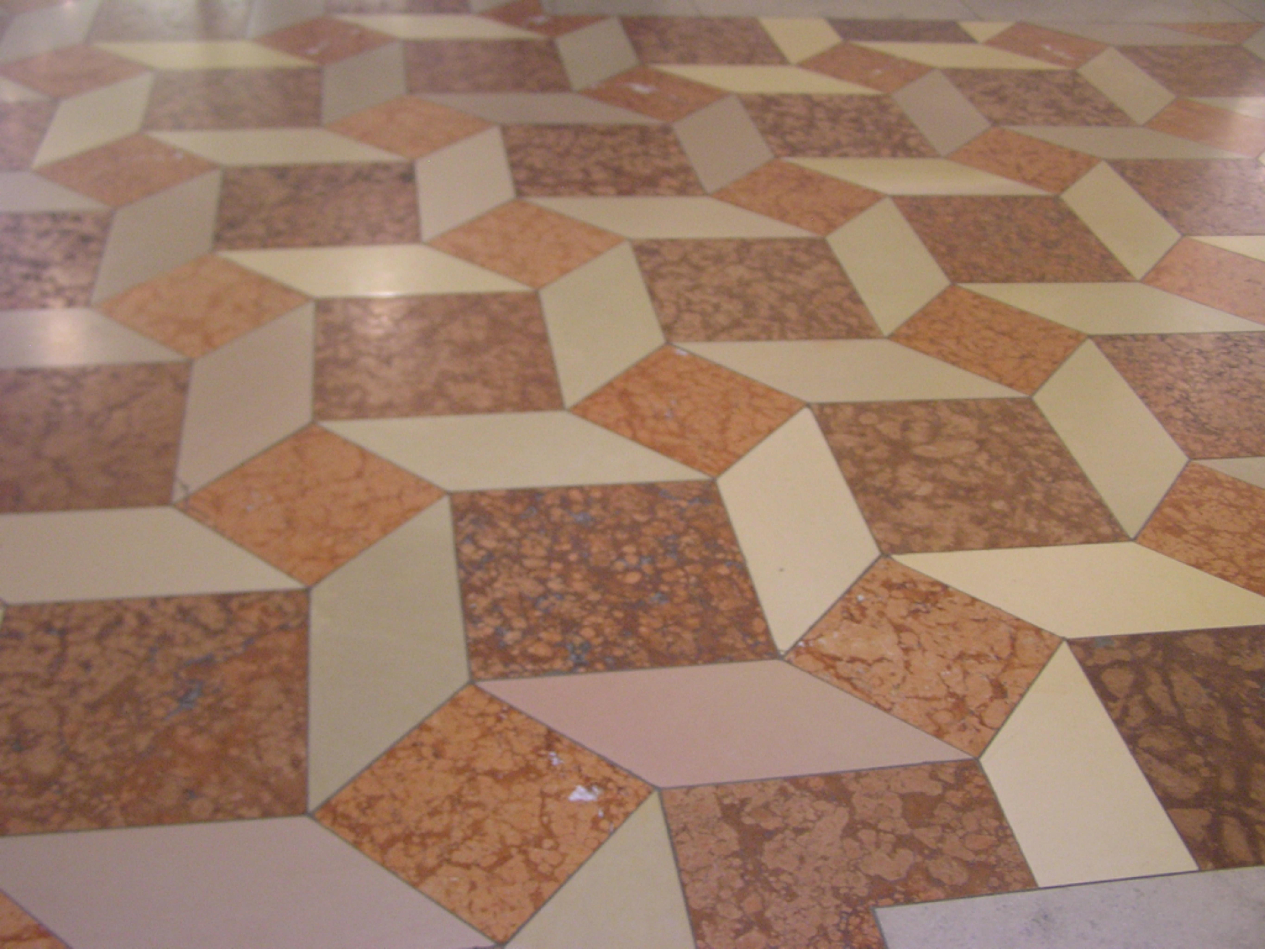
A neckbeard programmer gave me a book



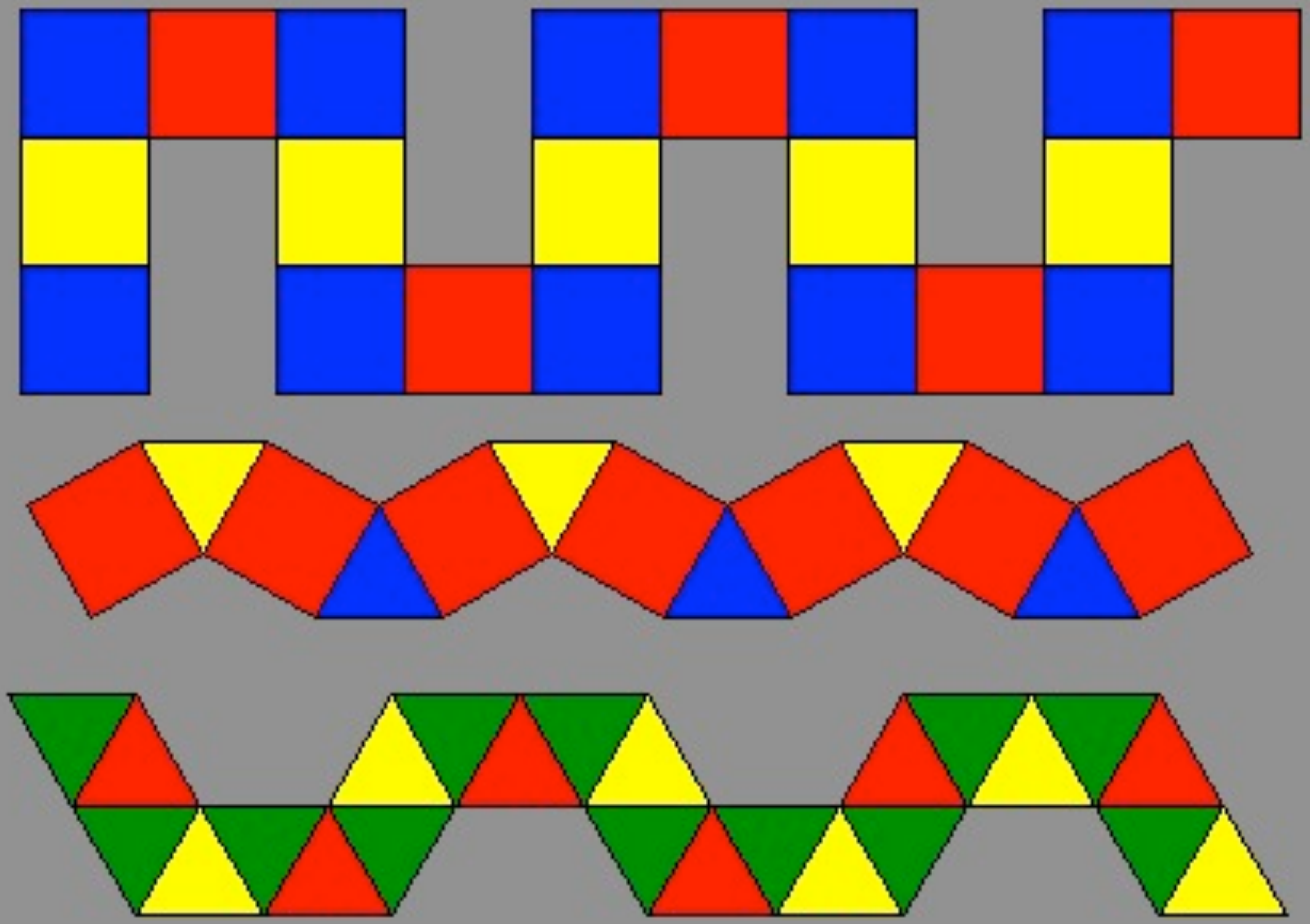
What are Design Patterns?
Time-tested reusable software architecture components. Many problems in software engineering occur time and again. If you've only gotten started in software development during the mobile application wave, then you may have only now started to notice patterns in your code, but haven't been able to pin point why things are familiar.

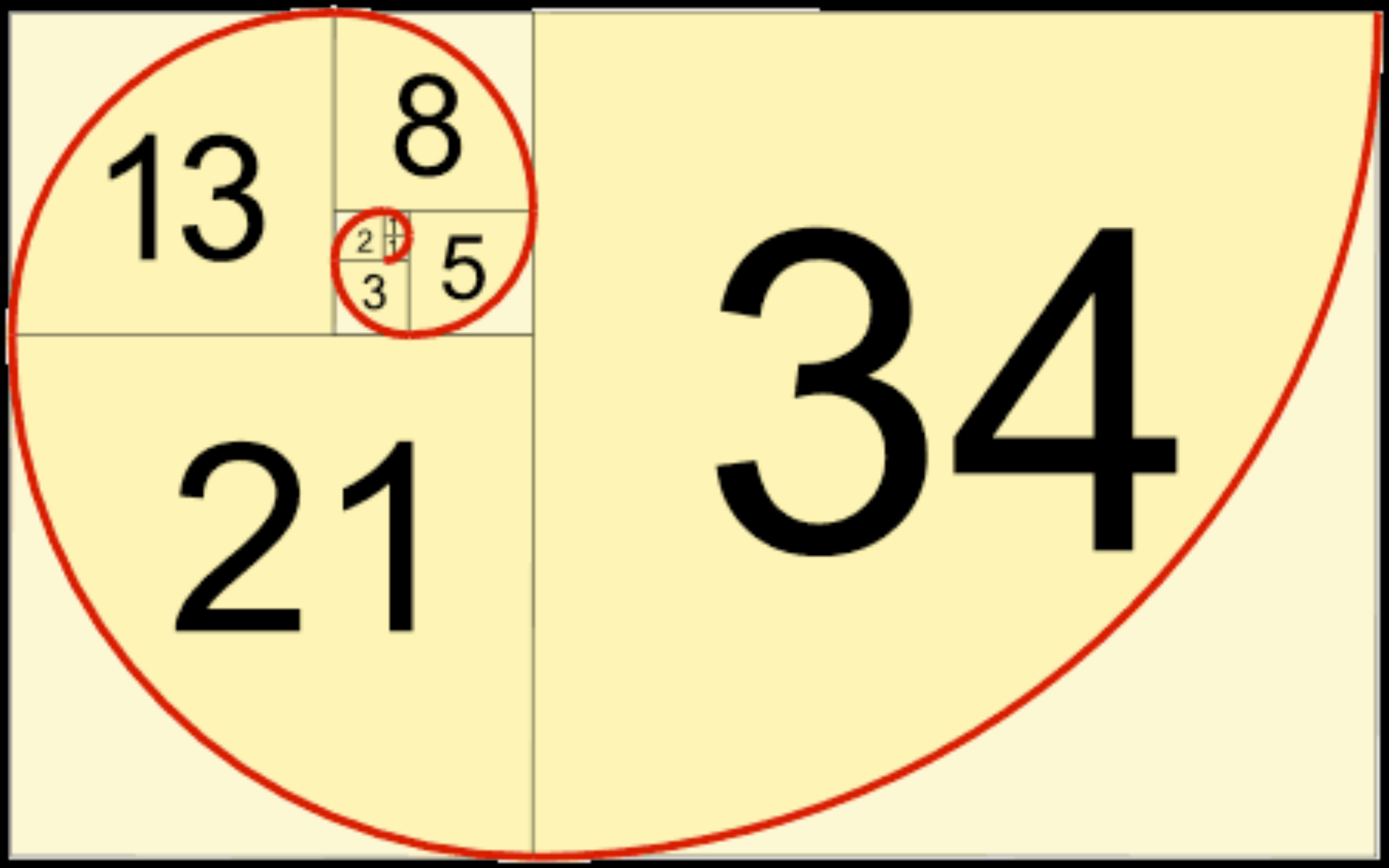
http://www.developer.com/design/article.php/1474561/What-Are-Design-Patterns-and-Do-I-Need-Them.htm

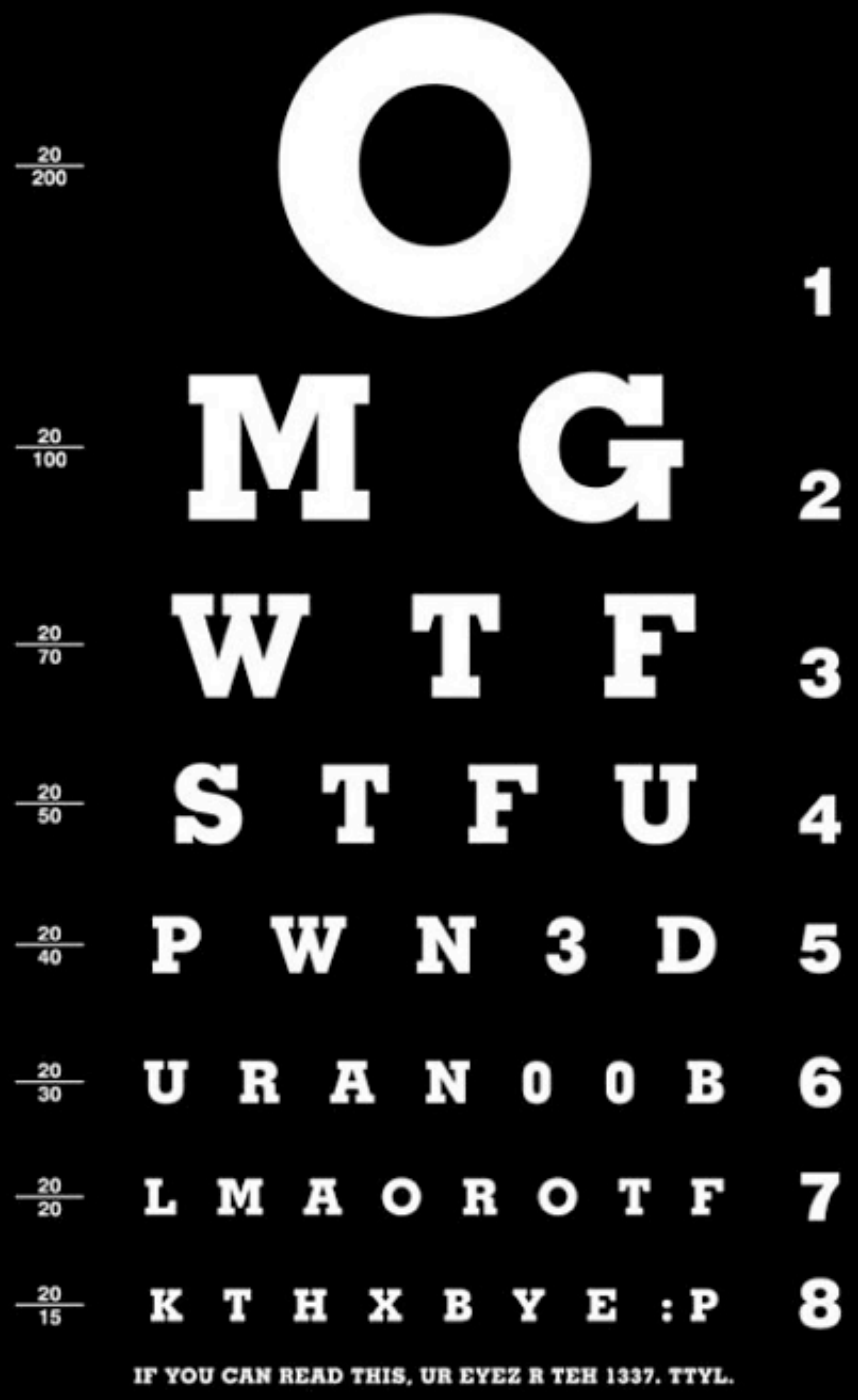
http://www.amazon.com/Design-Patterns-Elements-Reusable-Object-Oriented/dp/0201633612/ref=sr_1_1?ie=UTF8&qid=1363000537&sr=8-1&keywords=design+patterns



Wednesday, June 19, 13




13
8
2
1
1
3
5
21
34

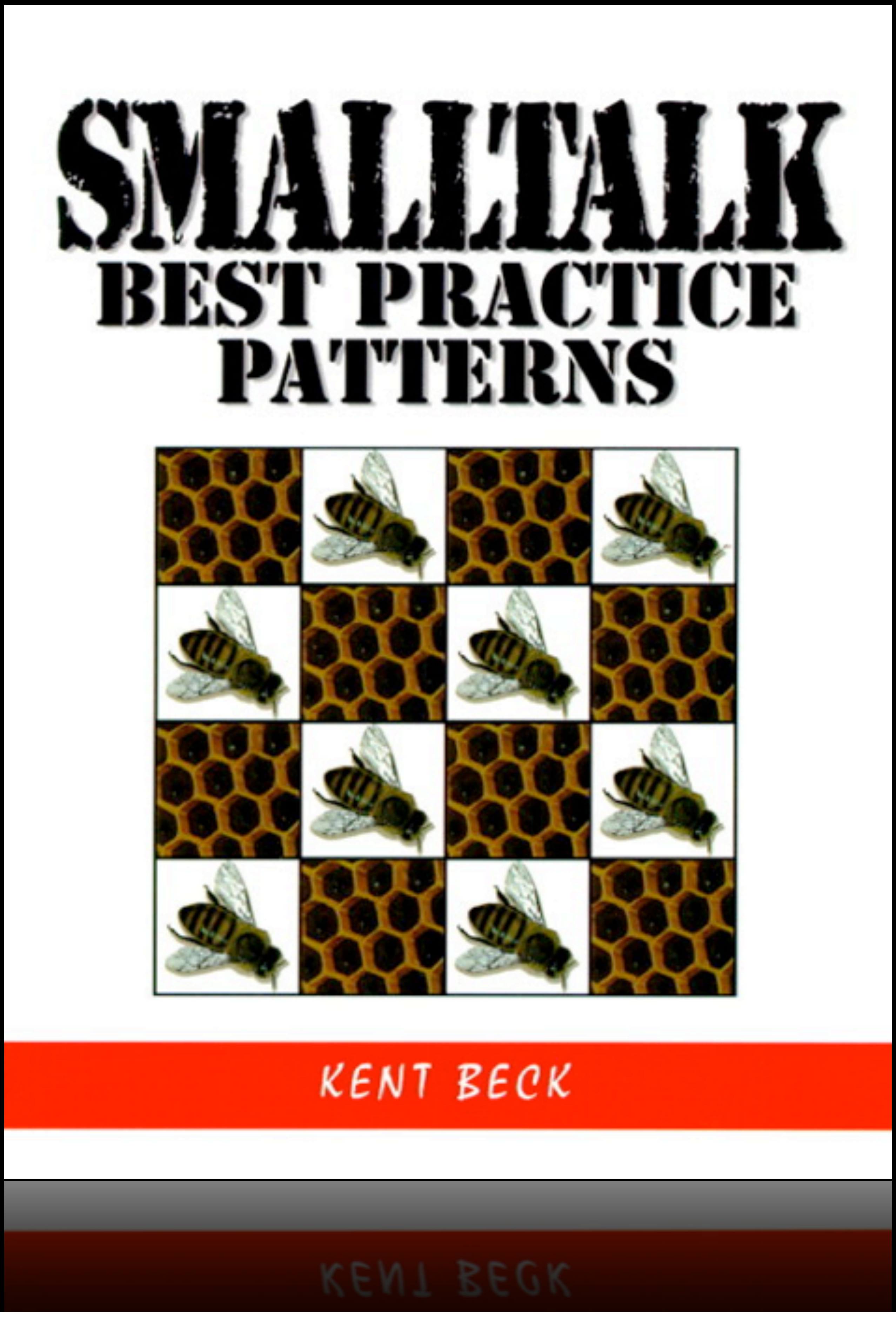

20/200 O 1
20/100 M G 2
20/70 W T F 3
20/50 S T F U 4
20/40 P W N 3 D 5
20/30 U R A N 0 0 B 6
20/20 L M A O R O T F 7
20/15 K T H X B Y E :P 8
IF YOU CAN READ THIS, UR EYEZ R TEH 1337. TTYL.

Wednesday, June 19, 13

Fractals, numbers and other patterns occur in nature. Our human brains have evolved to notice patterns. Patterns were first documented in building architecture. The idea was picked up by software engineers and then applied to software architecture.

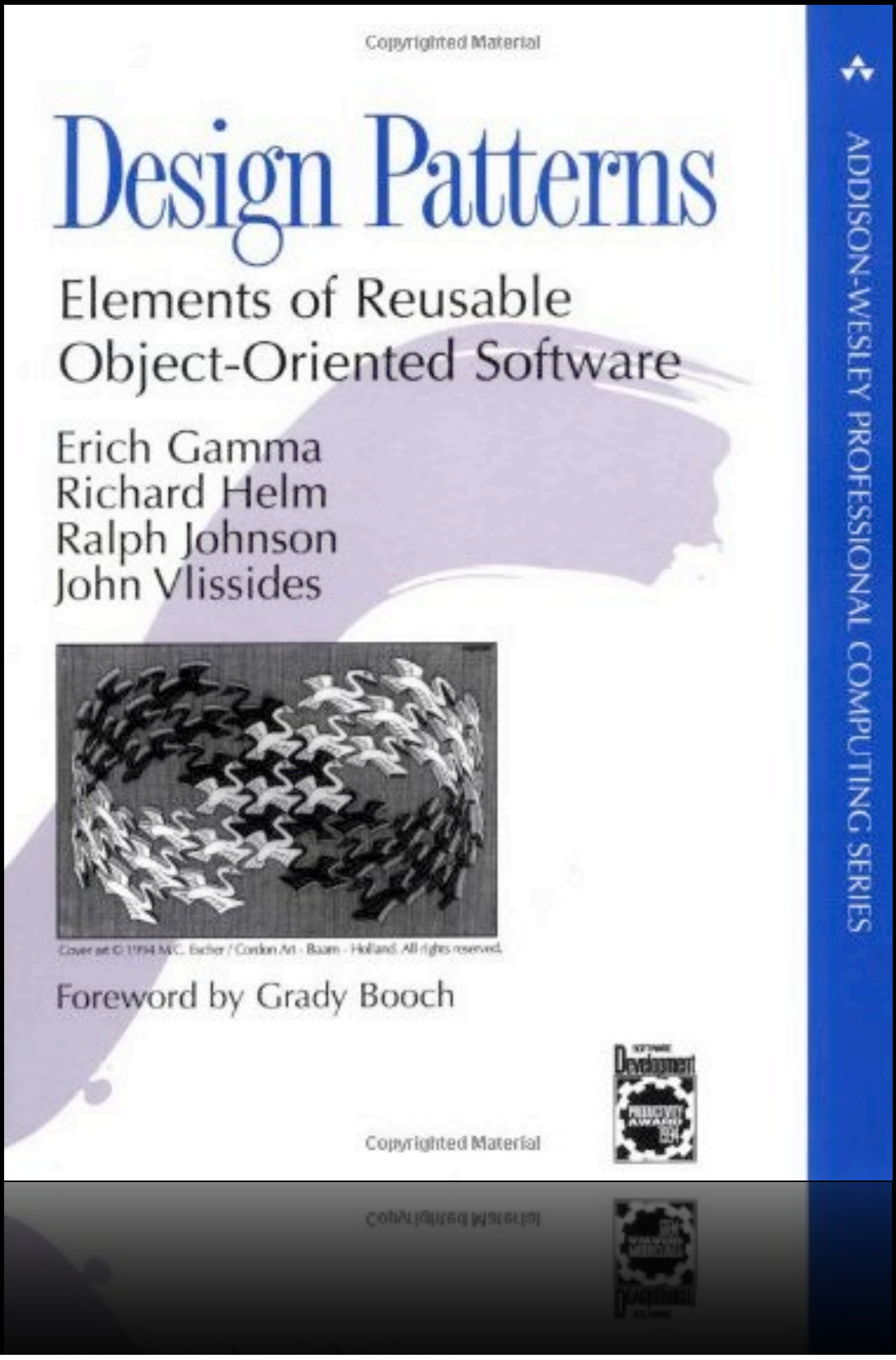
N

Wednesday, June 19, 13



More recently, on my journey/quest to become a better/master developer, I've been delving into books even older than the Design Patterns book. I've been reading SmallTalk Best Practices by Kent Beck.

I picked up this book to learn what previous masters knew. I found that over the course of my programming career, I had already encountered at least half of the code patterns in this book. I still had many to learn. But it made me realize that there is so much knowledge from previous generations of developers that, while not lost, is not implemented nearly enough in modern applications.



Copyrighted Material

Design Patterns

Elements of Reusable Object-Oriented Software

Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides

Cover art © 1994 M.C. Escher / Cordon Art - Baarn - Holland. All rights reserved.

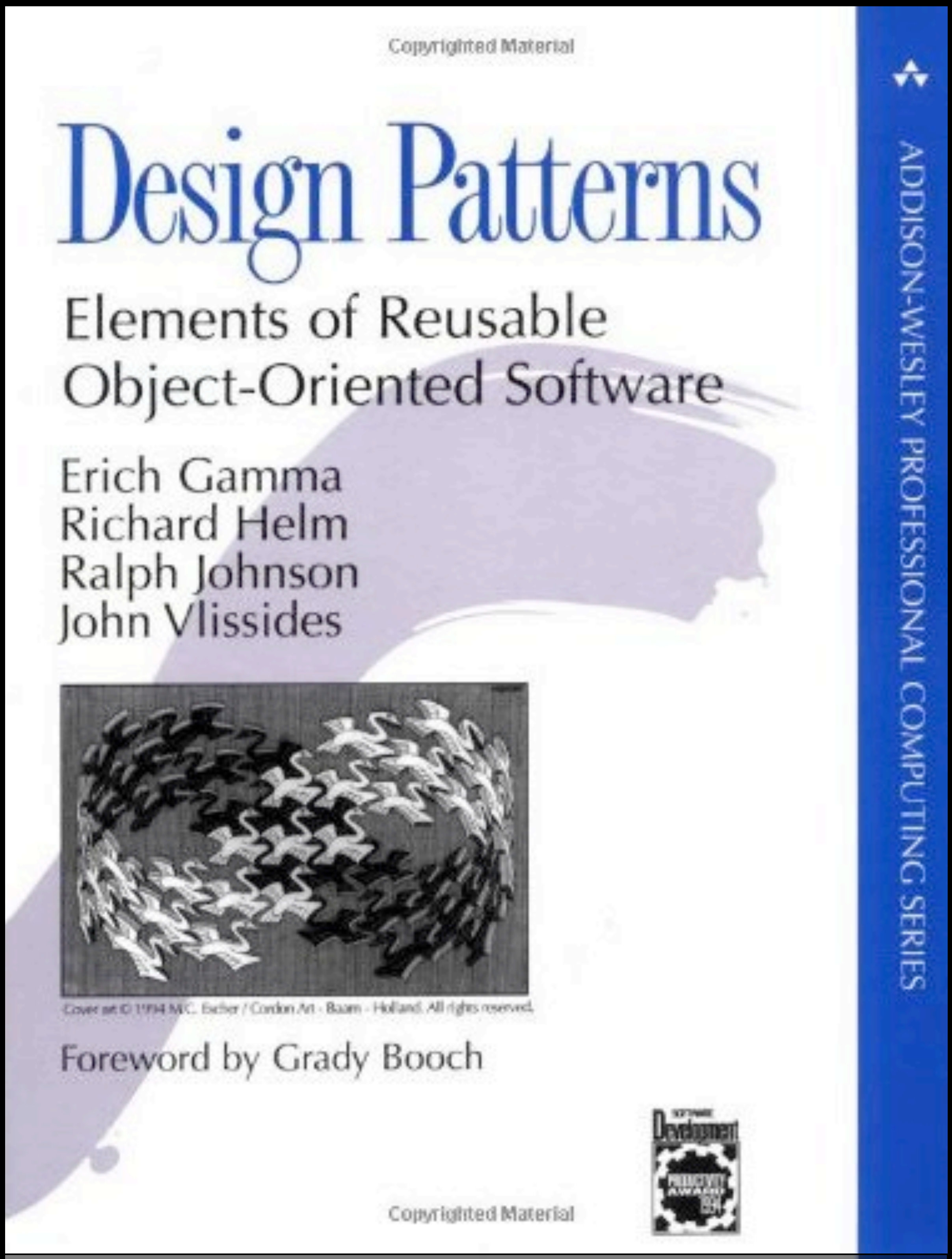
Foreword by Grady Booch

Copyrighted Material

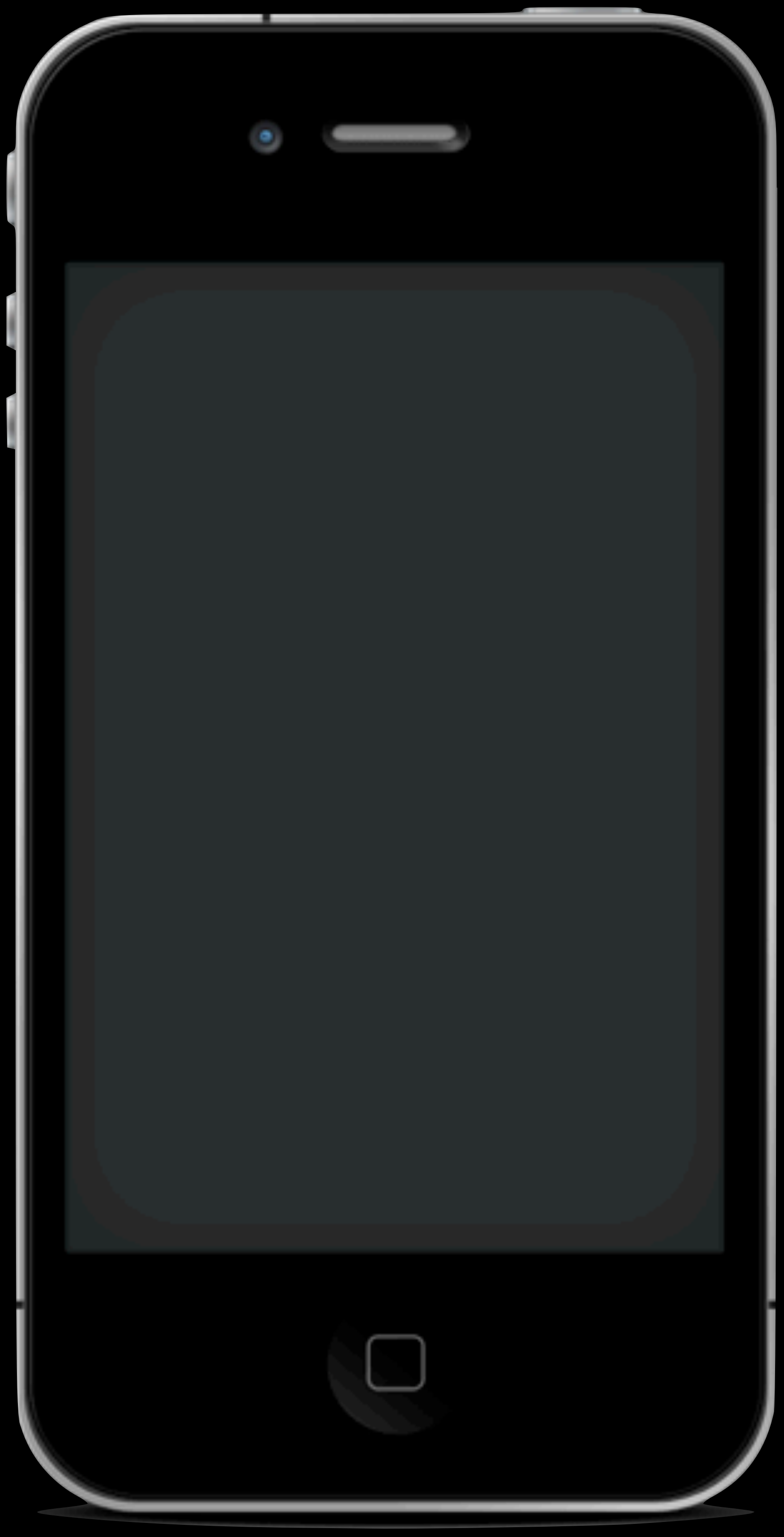
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

Wednesday, June 19, 13

And so I'm here today to share with you how patterns fit in this modern world of mobile apps.



+

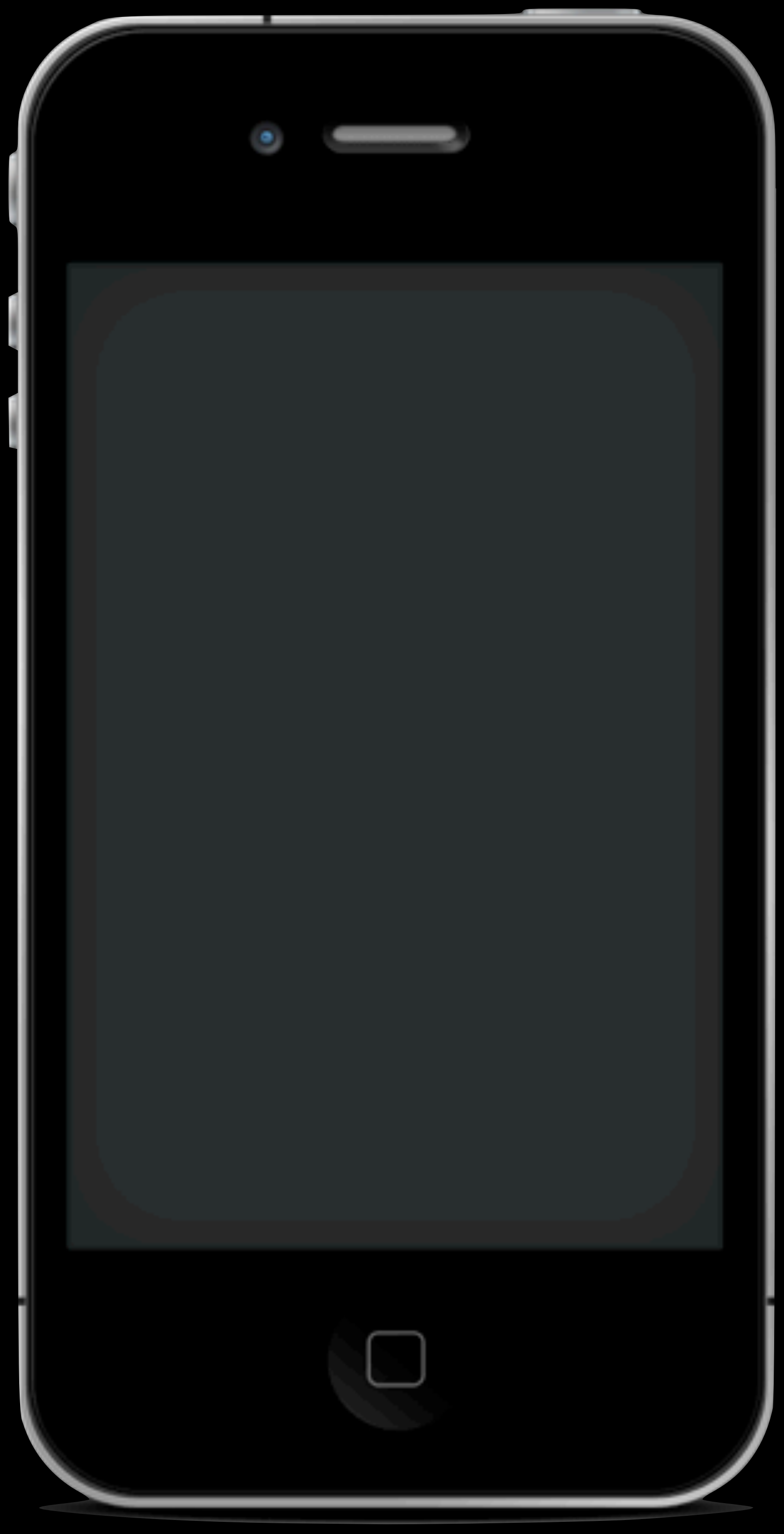


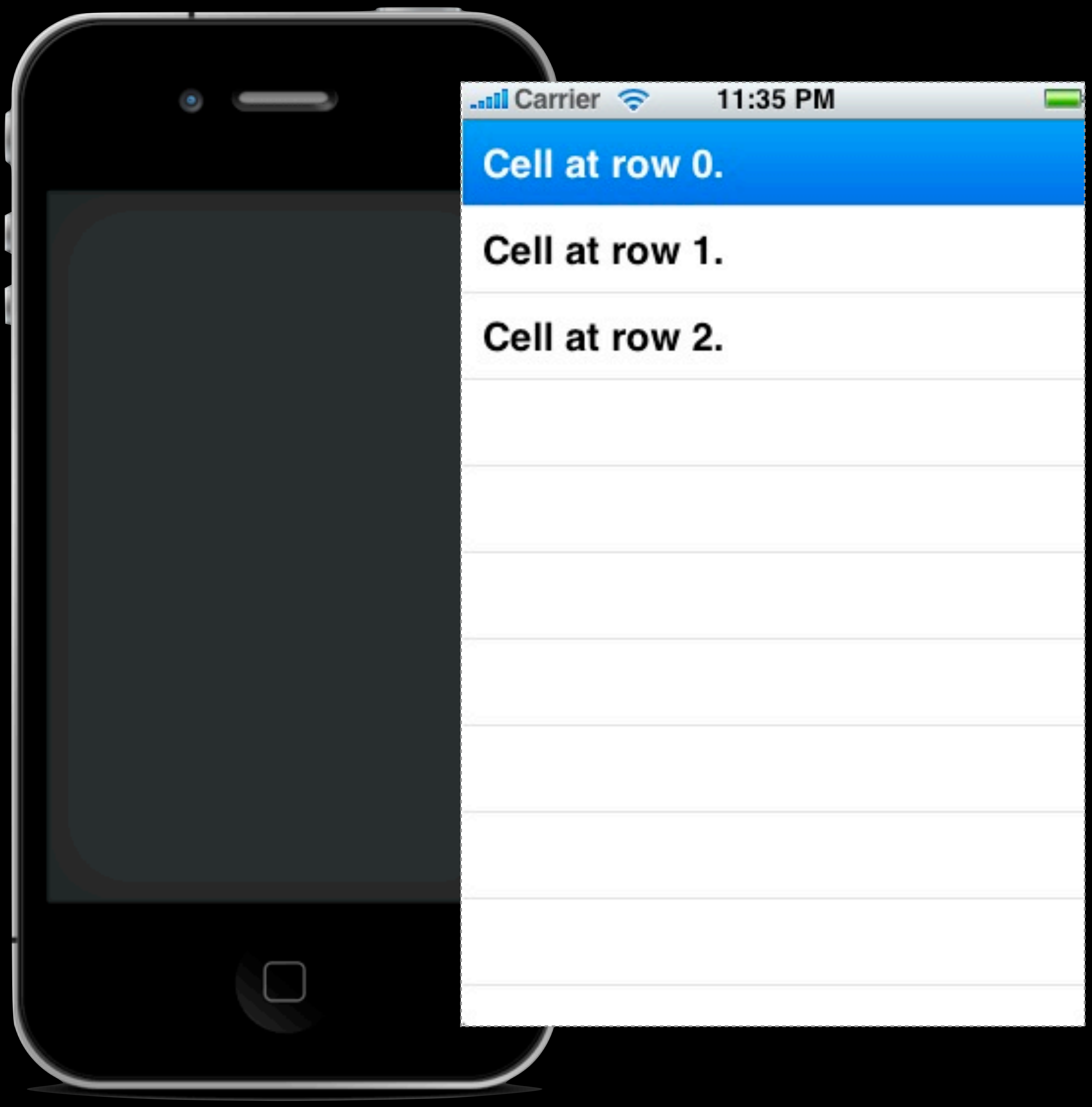
Wednesday, June 19, 13

Let's Build a Mobile App

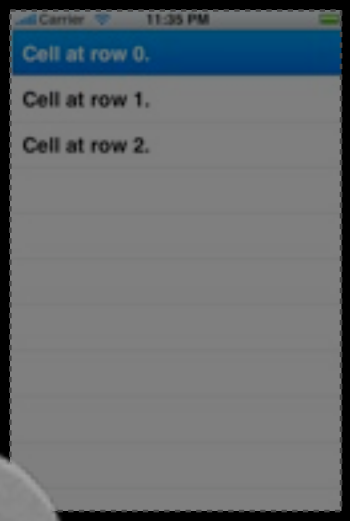
Wednesday, June 19, 13

The best way to explain these patterns, and when and where to use them in your apps is to build a mobile app of our own here.





Requirements
- Plugs into app.net
- Displays a list of posts
- Saves the tweets so relaunches are fast
- Can add new service API urls fast
- Doesn’t crash



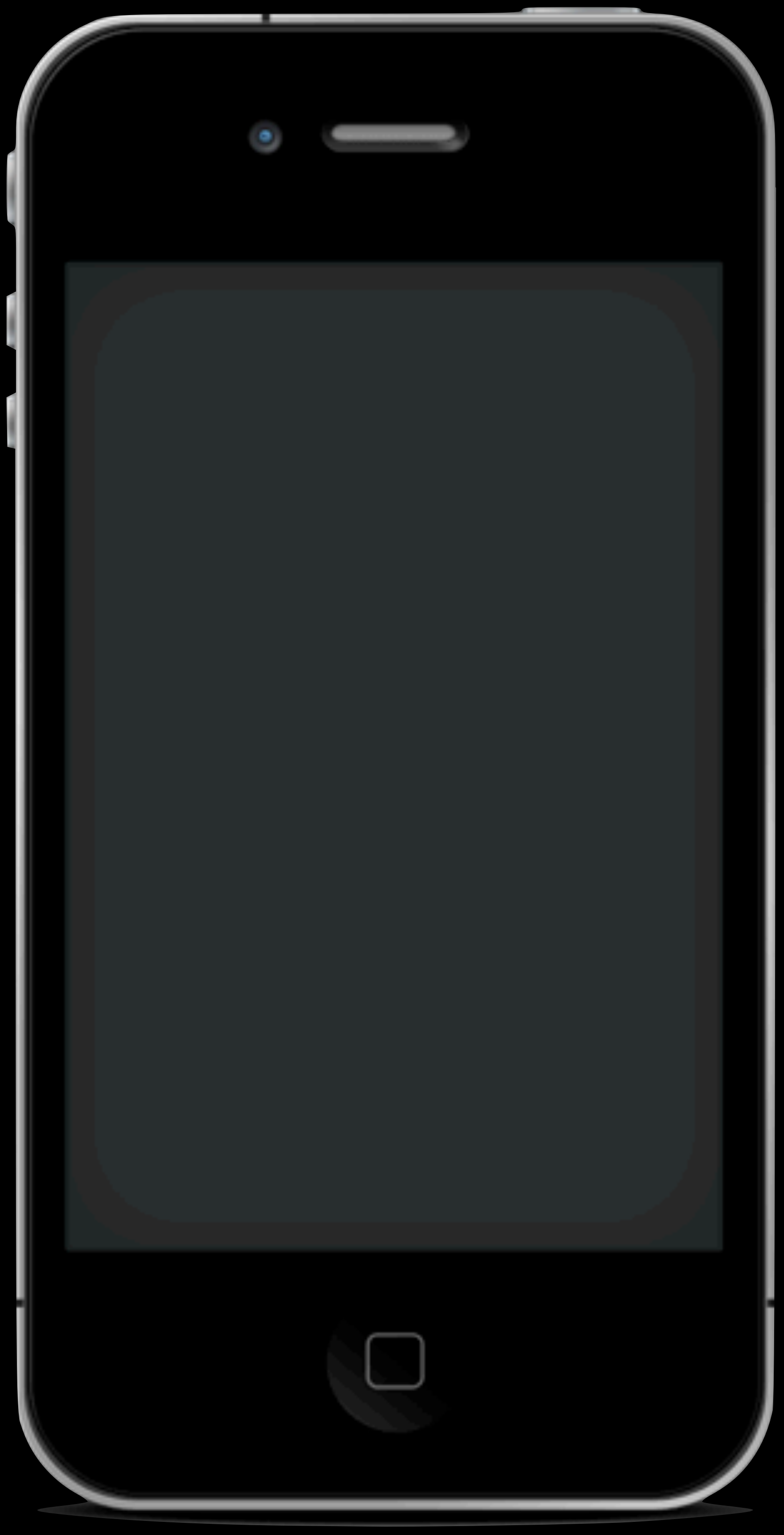
Wednesday, June 19, 13

Requirements
- Plugs into app.net
- Displays a list of posts
- Saves the tweets so relaunches are fast
- Can add new service API urls fast
- Doesn’t crash

Wednesday, June 19, 13

Requirements
- Plugs into app.net
- Displays a list of posts
- Saves the tweets so relaunches are fast
- Can add new service API urls fast
- Doesn’t crash

Requirements
- Plugs into app.net
- Displays a list of posts
- Saves the tweets so relaunches are fast
- Can add new service API urls fast
- Doesn't crash

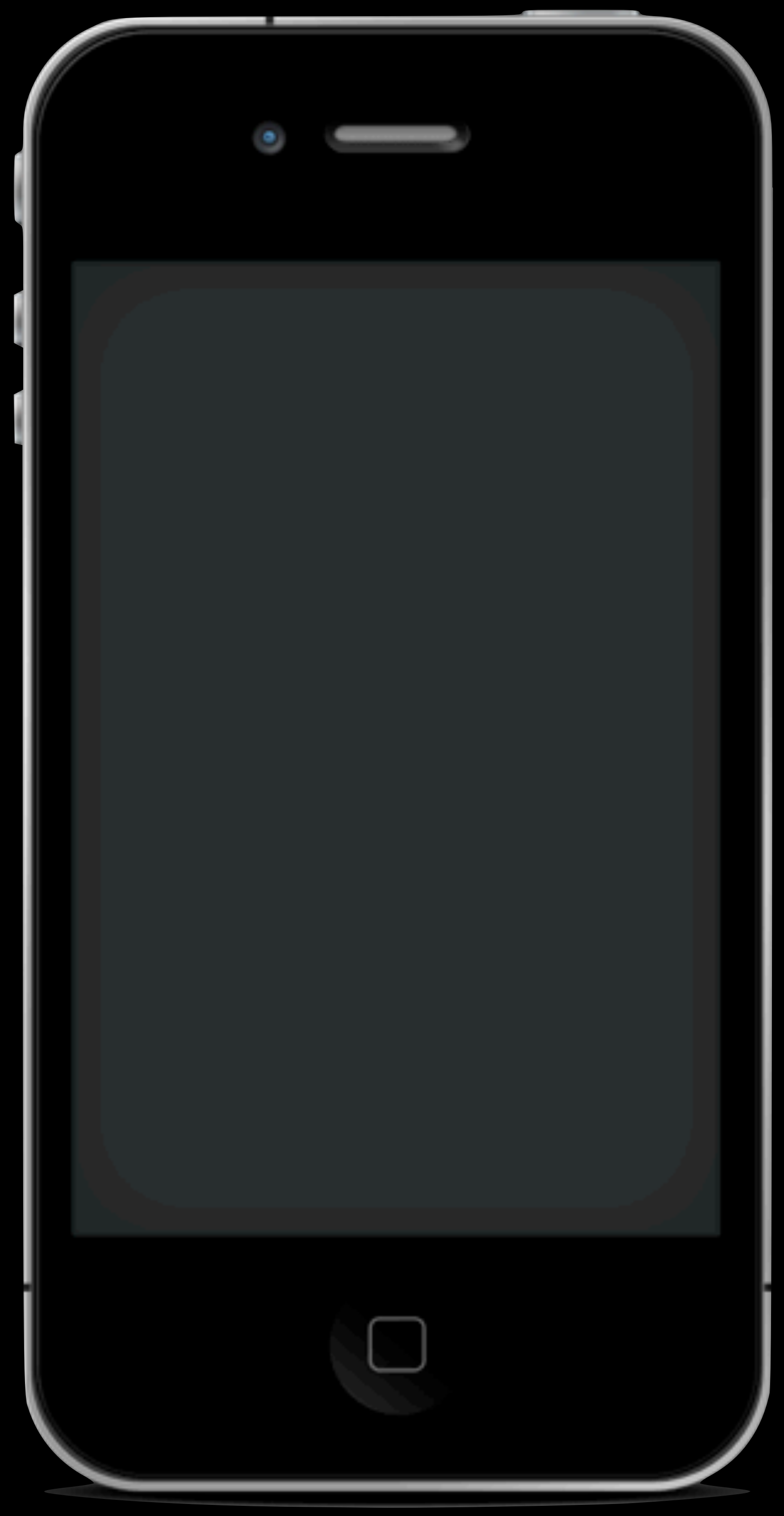


Cell at row 0.

Cell at row 1.

Cell at row 2.

Requirements
- Plugs into app.net
- Displays a list of posts
- Saves the tweets so relaunches are fast
- Can add new service API urls fast
- Doesn’t crash

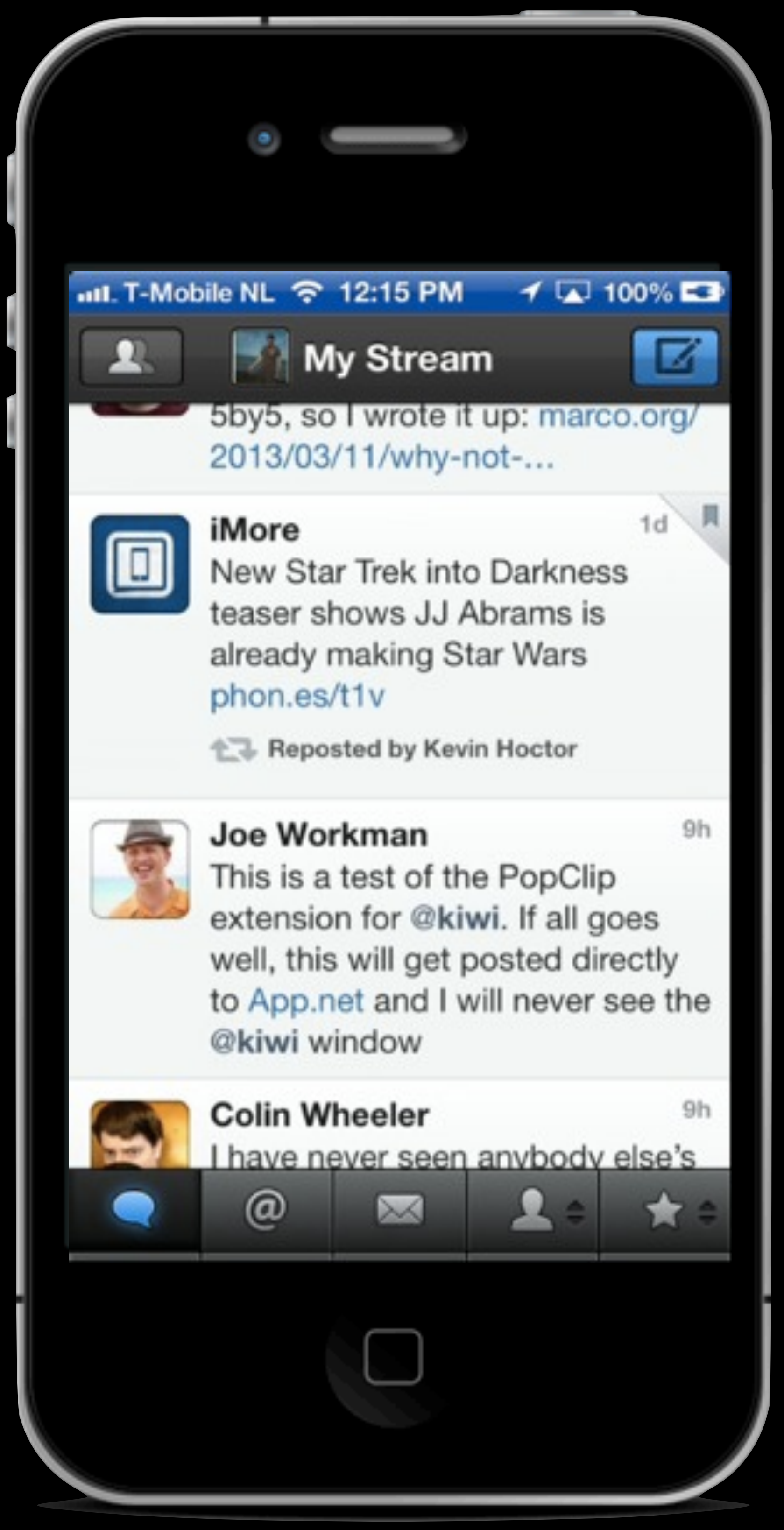


View

Controller

Model

Requirements
- Plugs into app.net
- Displays a list of posts
- Saves the tweets so relaunches are fast
- Can add new service API urls fast
- Doesn’t crash


T-Mobile NL 12:15 PM 100%

My Stream

5by5, so I wrote it up: marco.org/2013/03/11/why-not-...

iMore 1d
New Star Trek into Darkness teaser shows JJ Abrams is already making Star Wars phon.es/t1v
Reposted by Kevin Hoctor

Joe Workman 9h
This is a test of the PopClip extension for @kiwi. If all goes well, this will get posted directly to App.net and I will never see the @kiwi window

Colin Wheeler 9h
I have never seen anybody else's

Wednesday, June 19, 13

Requirements
- Plugs into app.net
- Displays a list of posts
- Saves the tweets so relaunches are fast
- Can add new service API urls fast
- Doesn't crash

MVC

Wednesday, June 19, 13

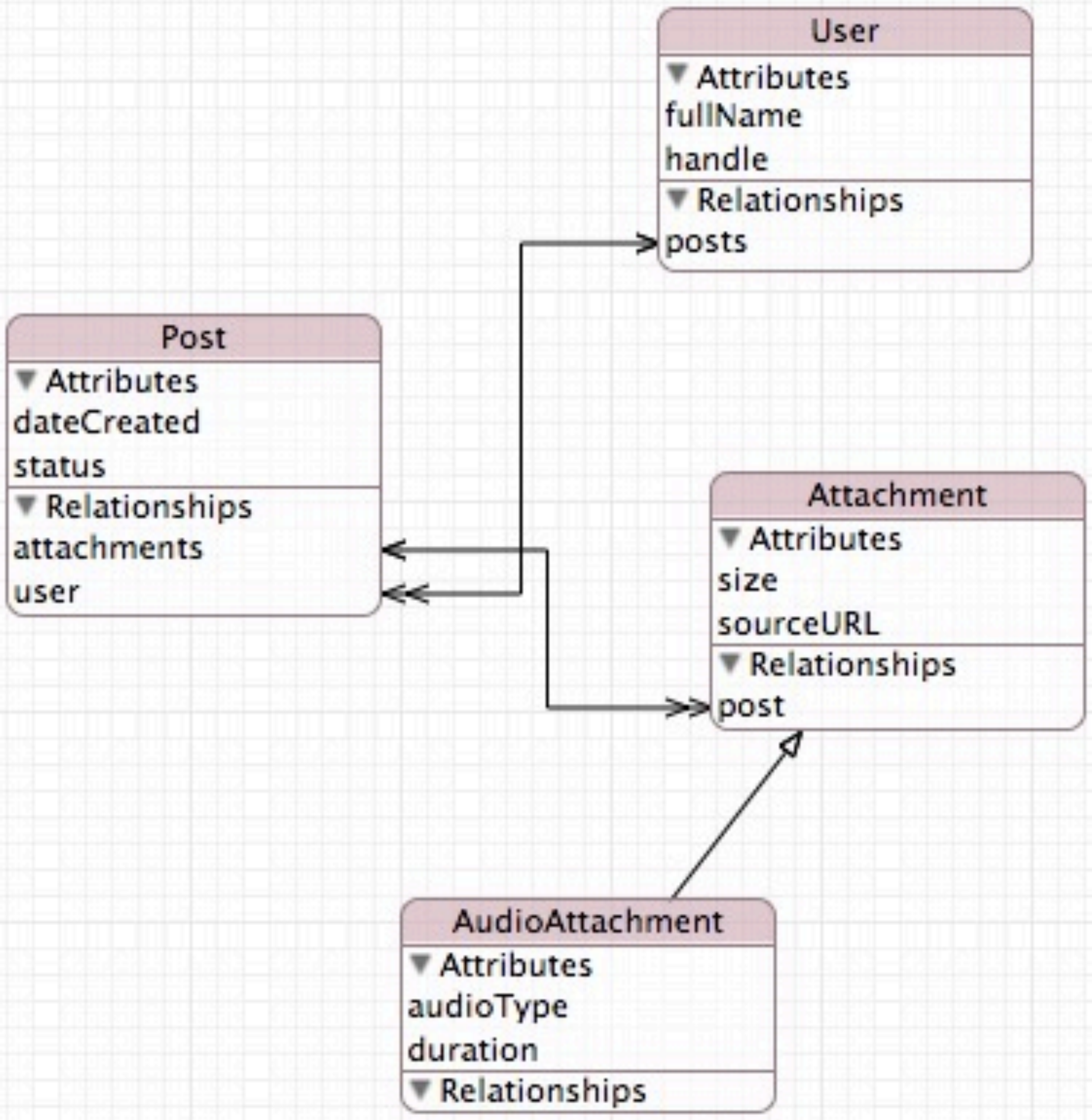
Let’s start with the mother of all patterns, MVC

Model View Controller

Model View Controller

Wednesday, June 19, 13

What goes in the model?

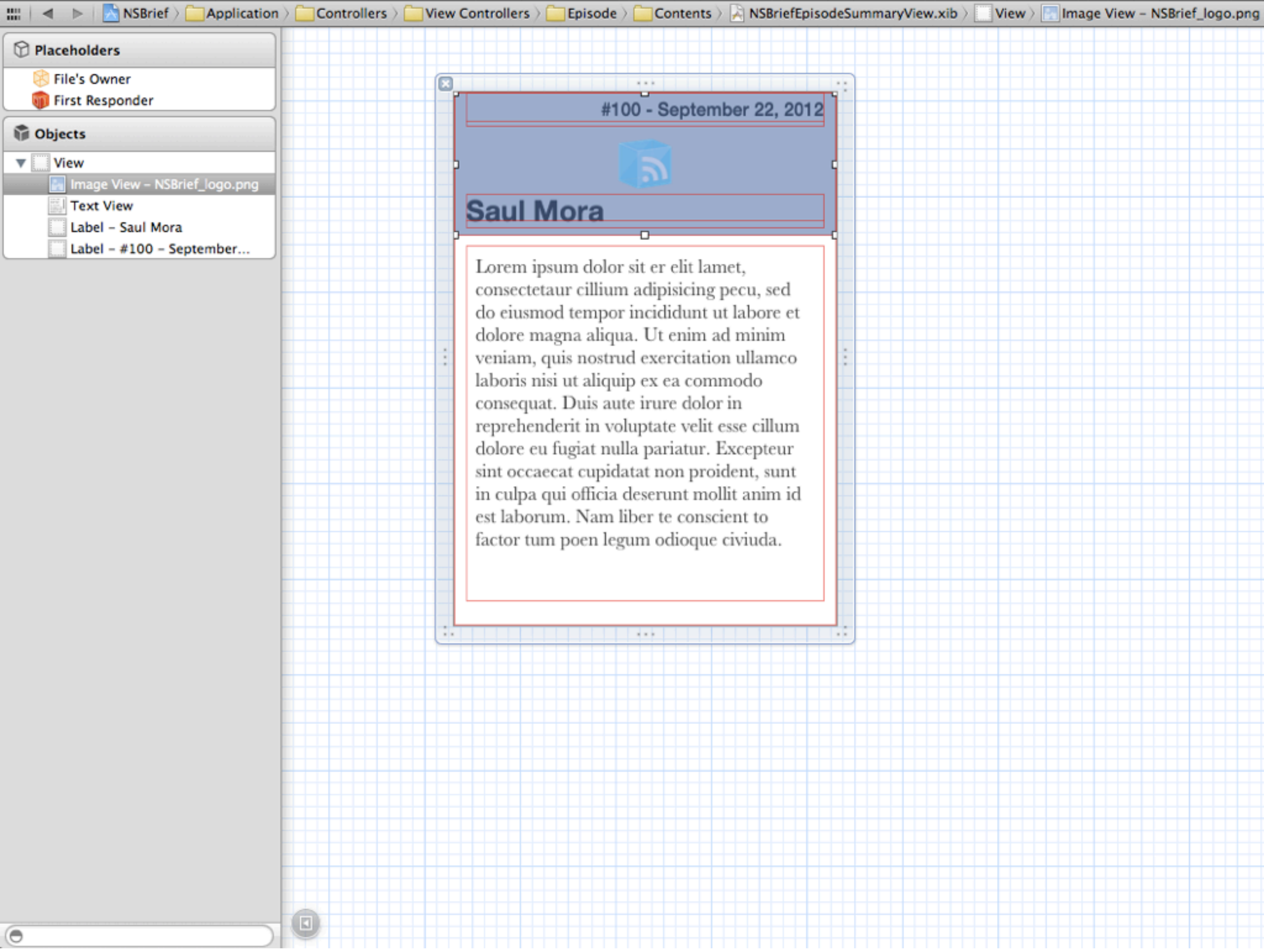

User
▼ Attributes
fullName
handle
▼ Relationships
posts
Post
▼ Attributes
dateCreated
status
▼ Relationships
attachments
user
Attachment
▼ Attributes
size
sourceURL
▼ Relationships
post
AudioAttachment
▼ Attributes
audioType
duration
▼ Relationships

```
@interface Post : _Post

+ (id) postWithId:(NSNumber *)postId;

- (void) downloadAttachments;

@end
```

Model View Controller

Wednesday, June 19, 13

What goes in a view?


NSBrief › Application › Controllers › View Controllers › Episode › Contents › NSBriefEpisodeSummaryView.xib › View › Image View – NSBrief_logo.png
Placeholders
File's Owner
First Responder
Objects
View
Image View – NSBrief_logo.png
Text View
Label – Saul Mora
Label – #100 – September...

#100 - September 22, 2012

Saul Mora

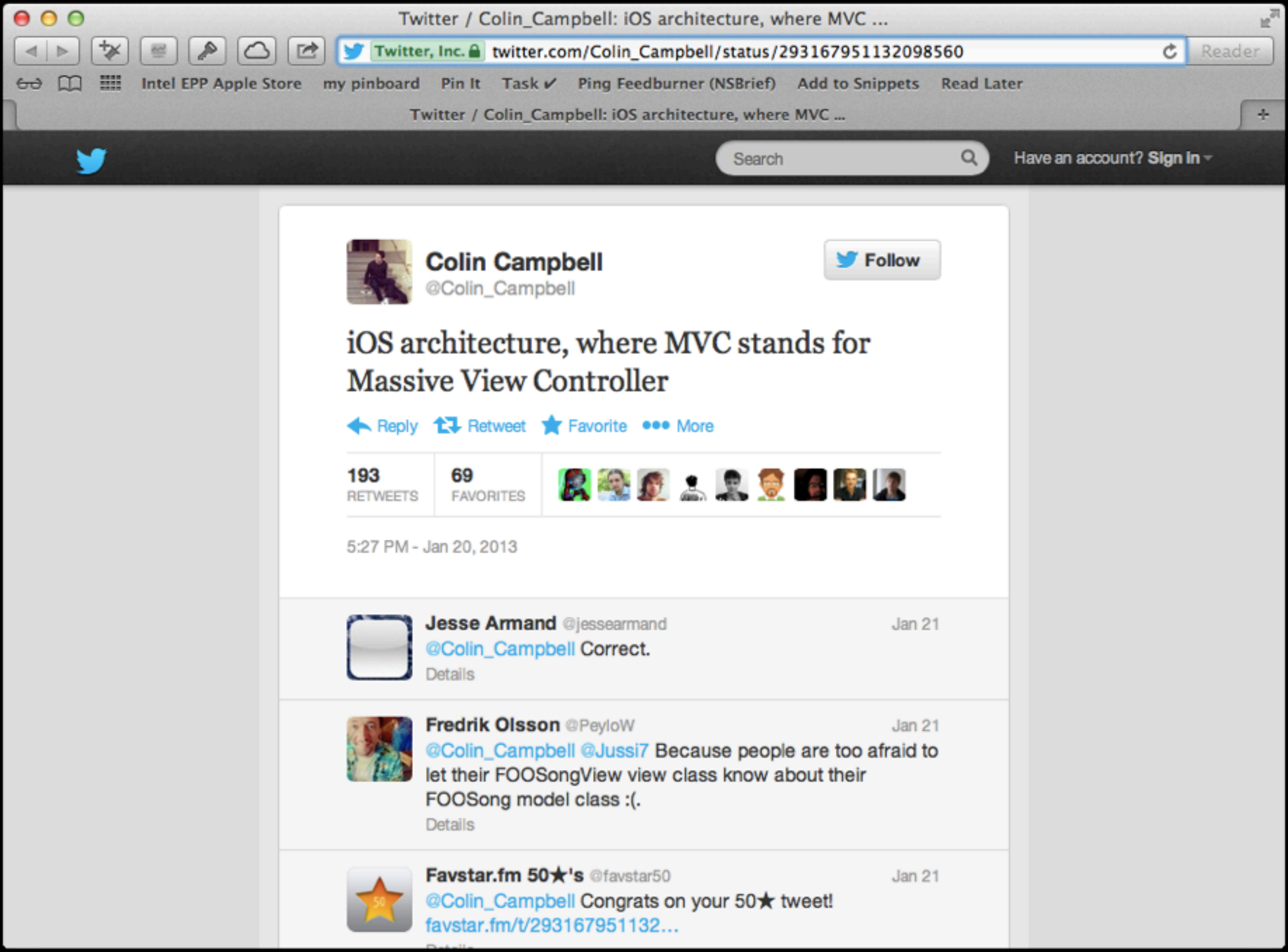
Lorem ipsum dolor sit er elit lamet, consectetaur cillium adipisicing pecu, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. Nam liber te conscient to factor tum poen legum odioque civiuda.

Model View Controller

Wednesday, June 19, 13

What goes in a Controller?

?


Twitter / Colin_Campbell: iOS architecture, where MVC ...
twitter.com/Colin_Campbell/status/293167951132098560
Intel EPP Apple Store
my pinboard
Pin It
Task ✔
Ping Feedburner (NSBrief)
Add to Snippets
Read Later
Search
Have an account? Sign in
Follow
Reply
Retweet
Favorite
More

Colin Campbell @Colin_Campbell

iOS architecture, where MVC stands for Massive View Controller

193 RETWEETS 69 FAVORITES

5:27 PM - Jan 20, 2013

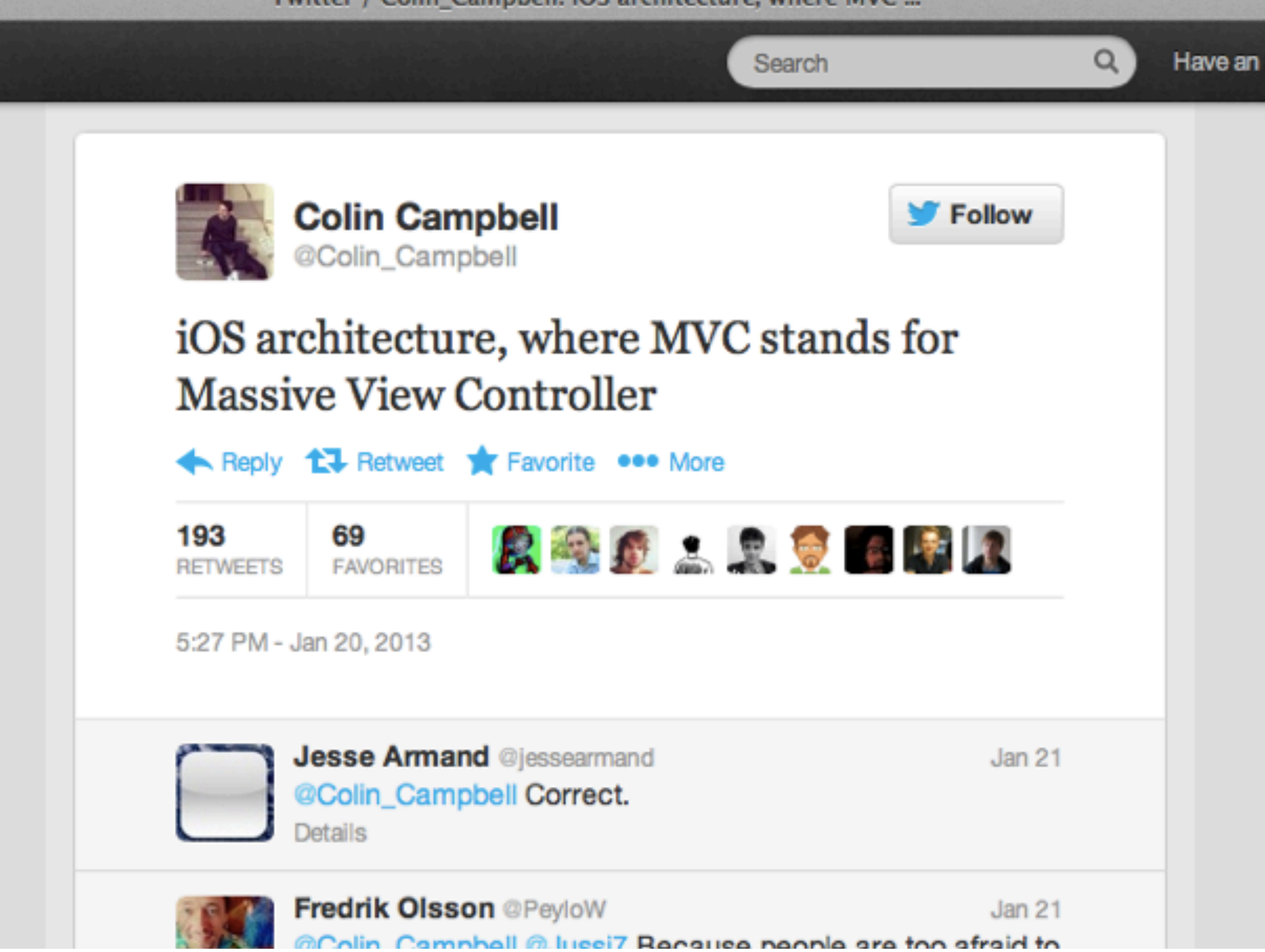
Jesse Armand @jessearmand Jan 21
@Colin_Campbell Correct.
Details

Fredrik Olsson @PeyloW Jan 21
@Colin_Campbell @Jussi7 Because people are too afraid to let their FOOSongView view class know about their FOOSong model class :(.
Details

Favstar.fm 50★'s @favstar50 Jan 21
@Colin_Campbell Congrats on your 50★ tweet! favstar.fm/t/293167951132...

Wednesday, June 19, 13

Massive View Controller


Colin Campbell
@Colin_Campbell

iOS architecture, where MVC stands for Massive View Controller

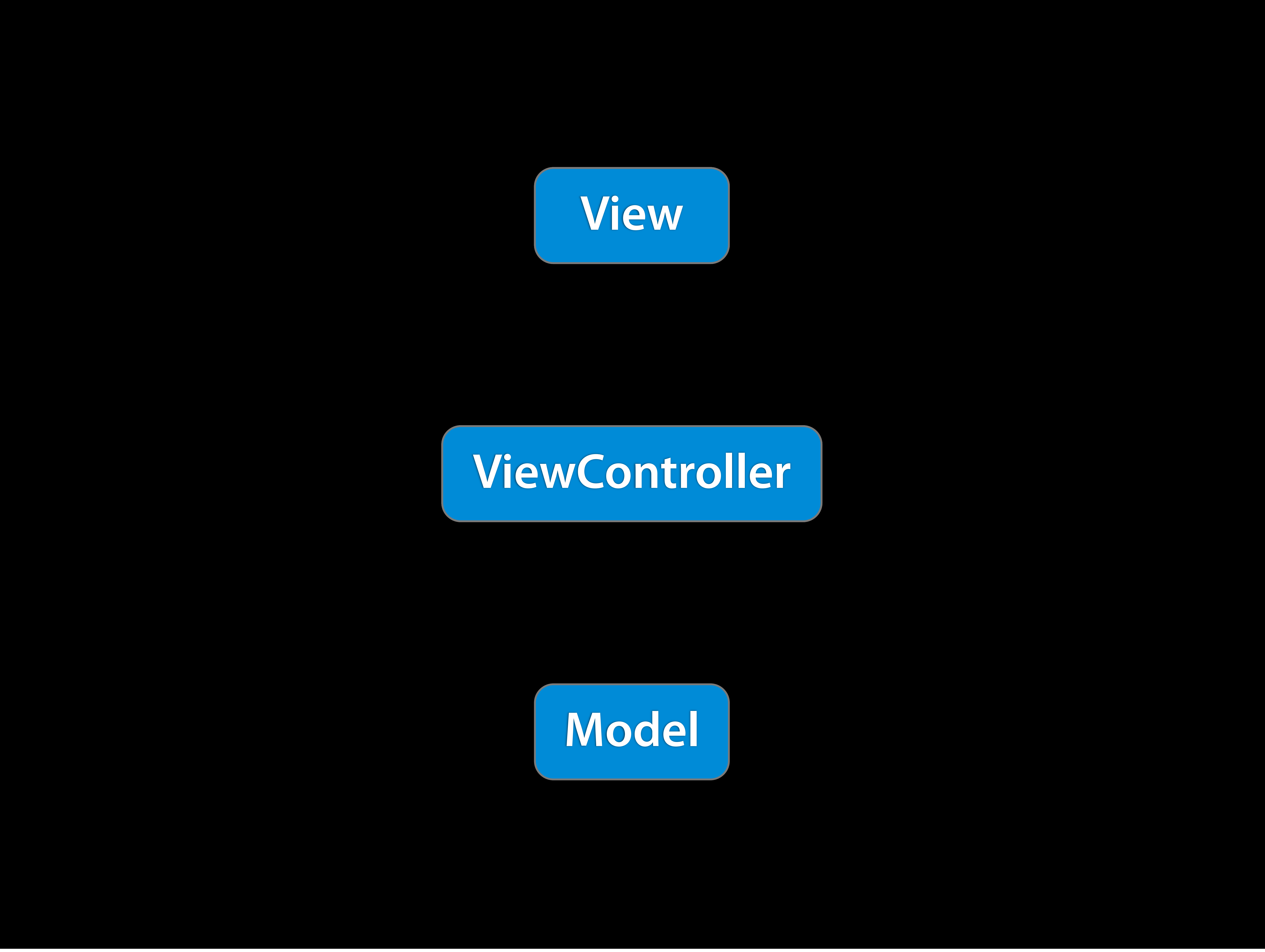
193 RETWEETS 69 FAVORITES

5:27 PM - Jan 20, 2013

Jesse Armand @jessearmand Jan 21
@Colin_Campbell Correct.

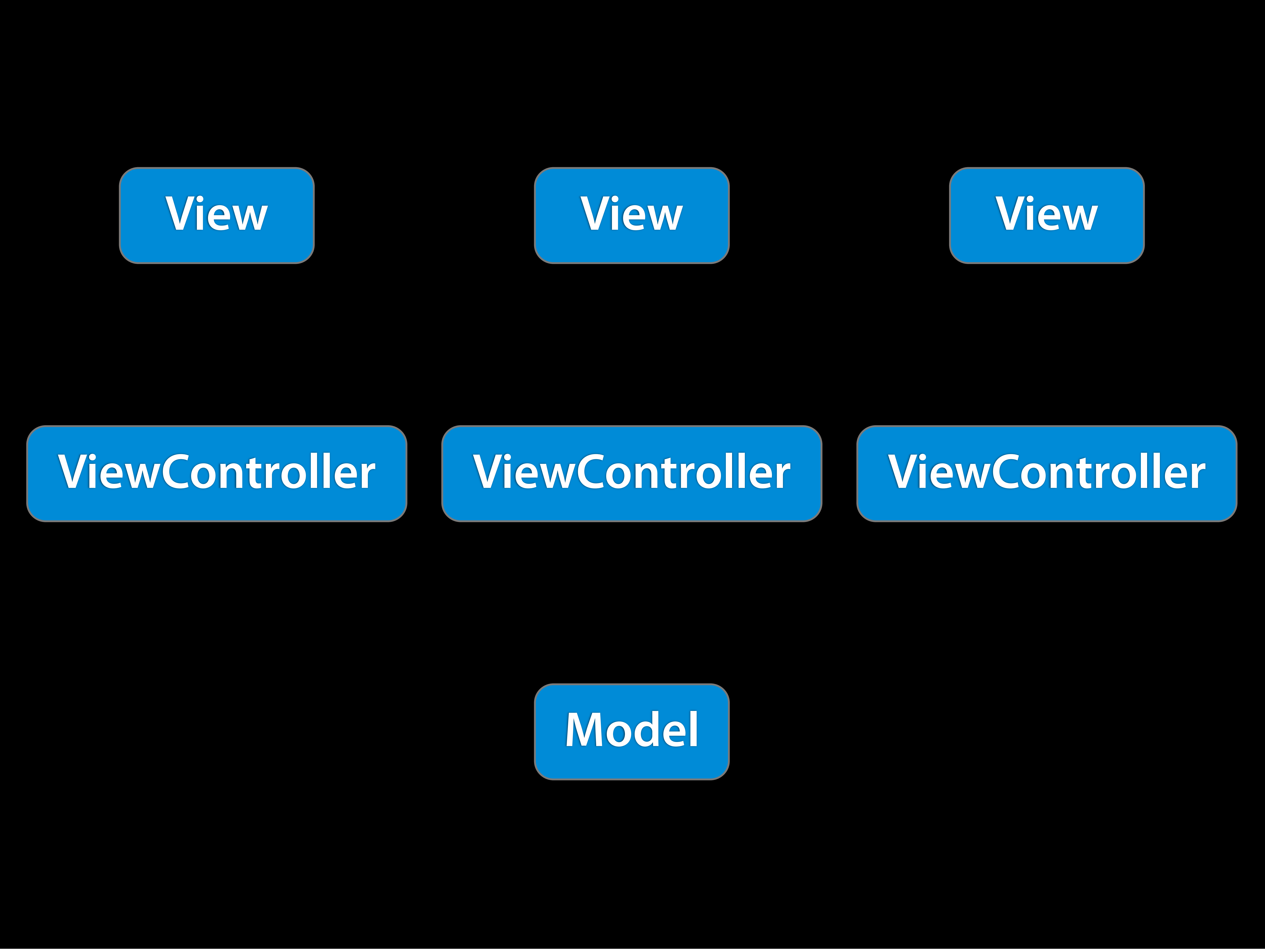
Fredrik Olsson @PeyloW Jan 21

Massive View Controller

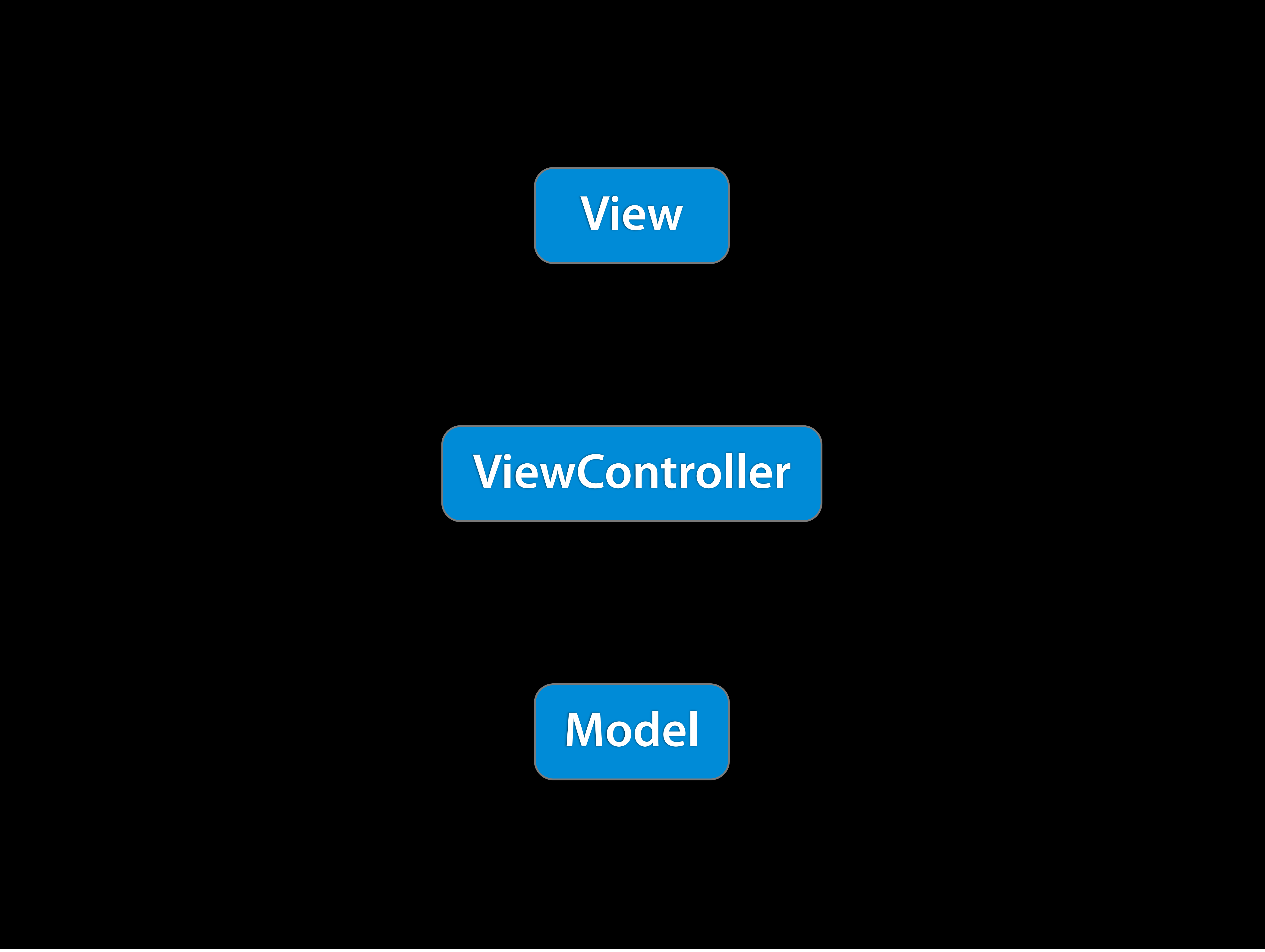


Wednesday, June 19, 13

On iOS and Mac, the concept of ViewControllers are a tad different

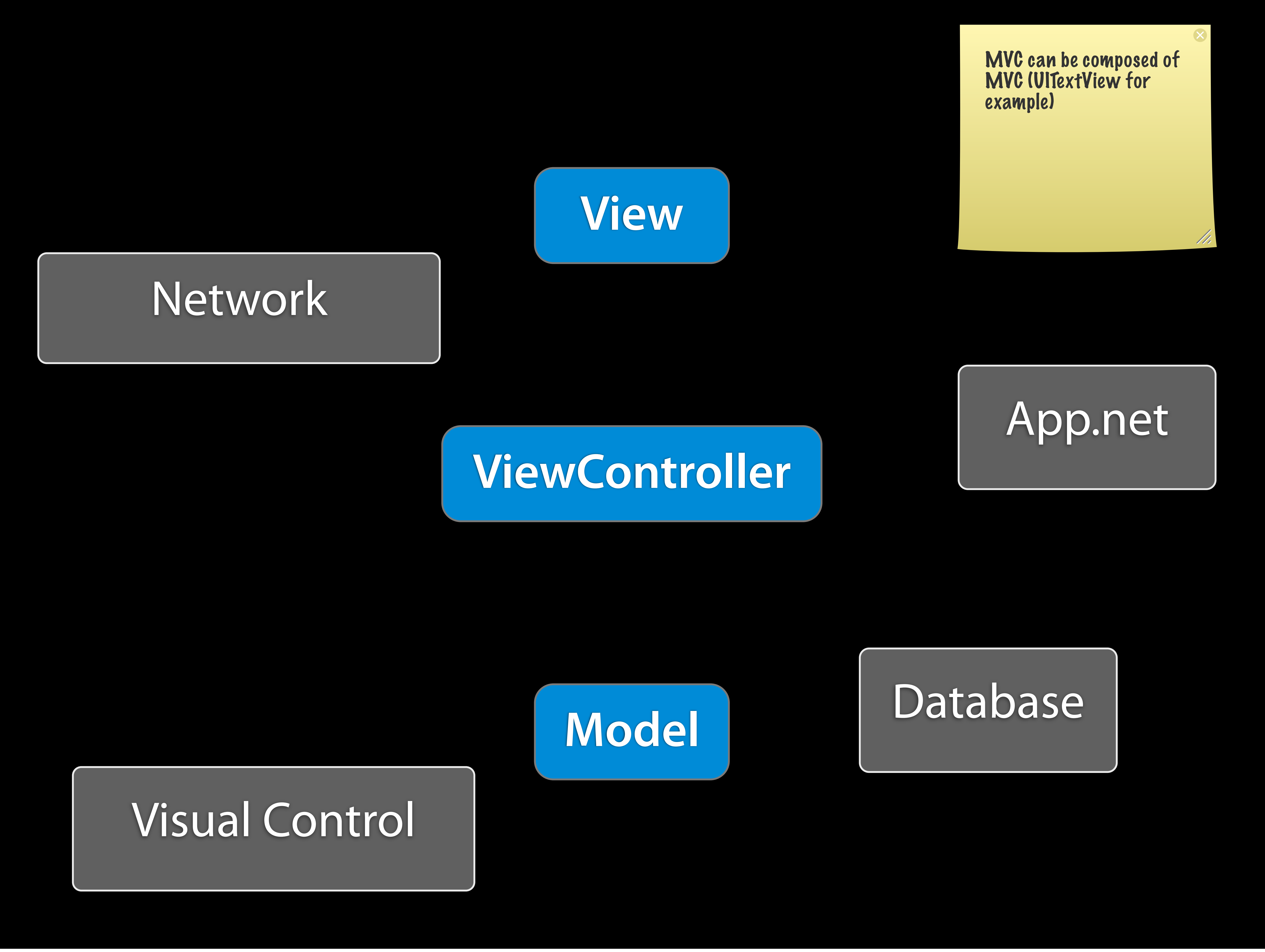


Ideally, your viewControllers should be able to display the same model information in multiple ways.



Wednesday, June 19, 13

How do they all work together?

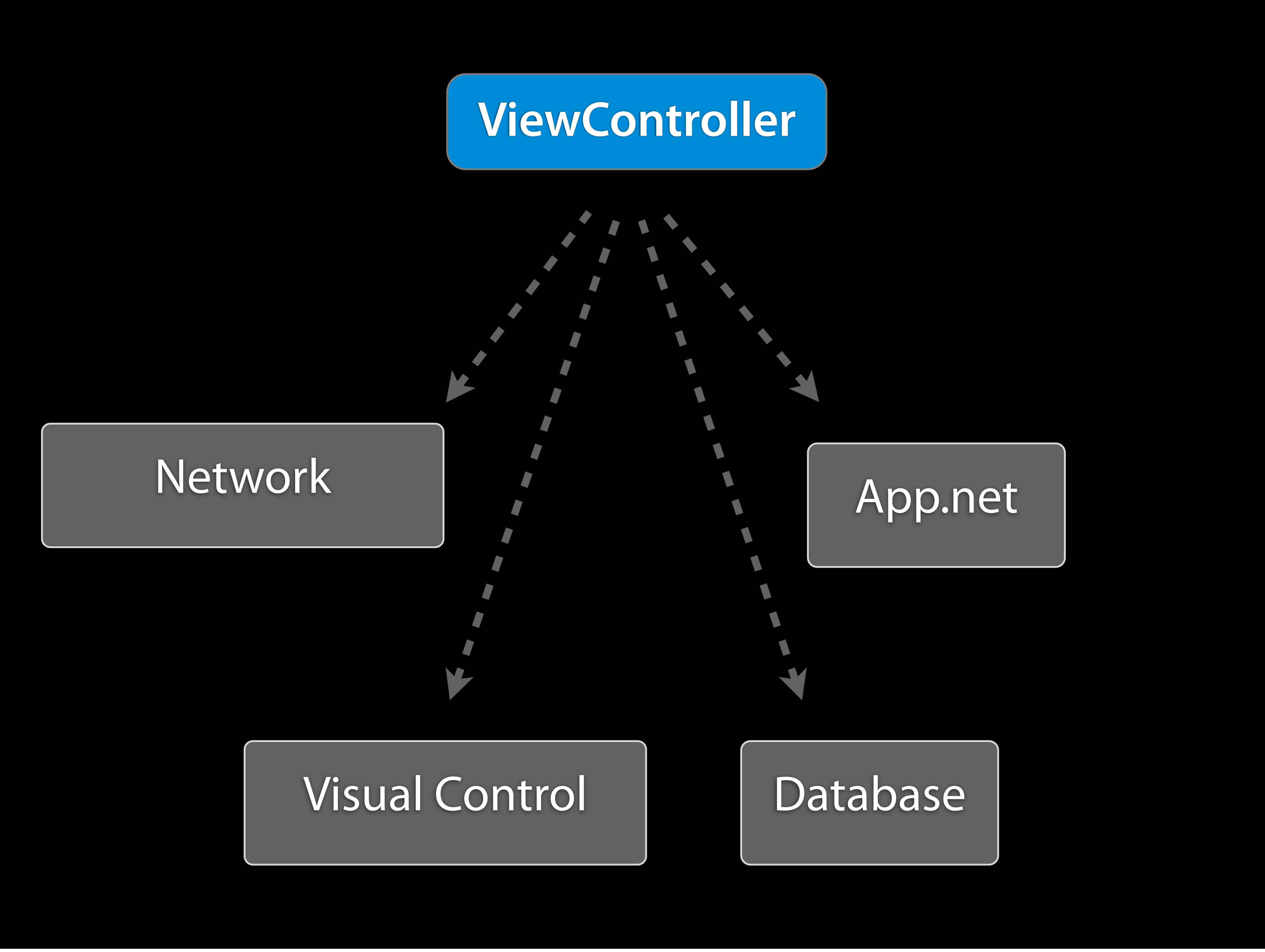


Wednesday, June 19, 13

Q: What goes in a viewcontroller?
A: All the things needed to present data into the view

Rule of Thumb: If it involves networking code, database code, drawing code, algorithms, it probably doesn't go in here.

The view controller should connect pieces, but not contain the logic for all those things...like a network stack, or a database...

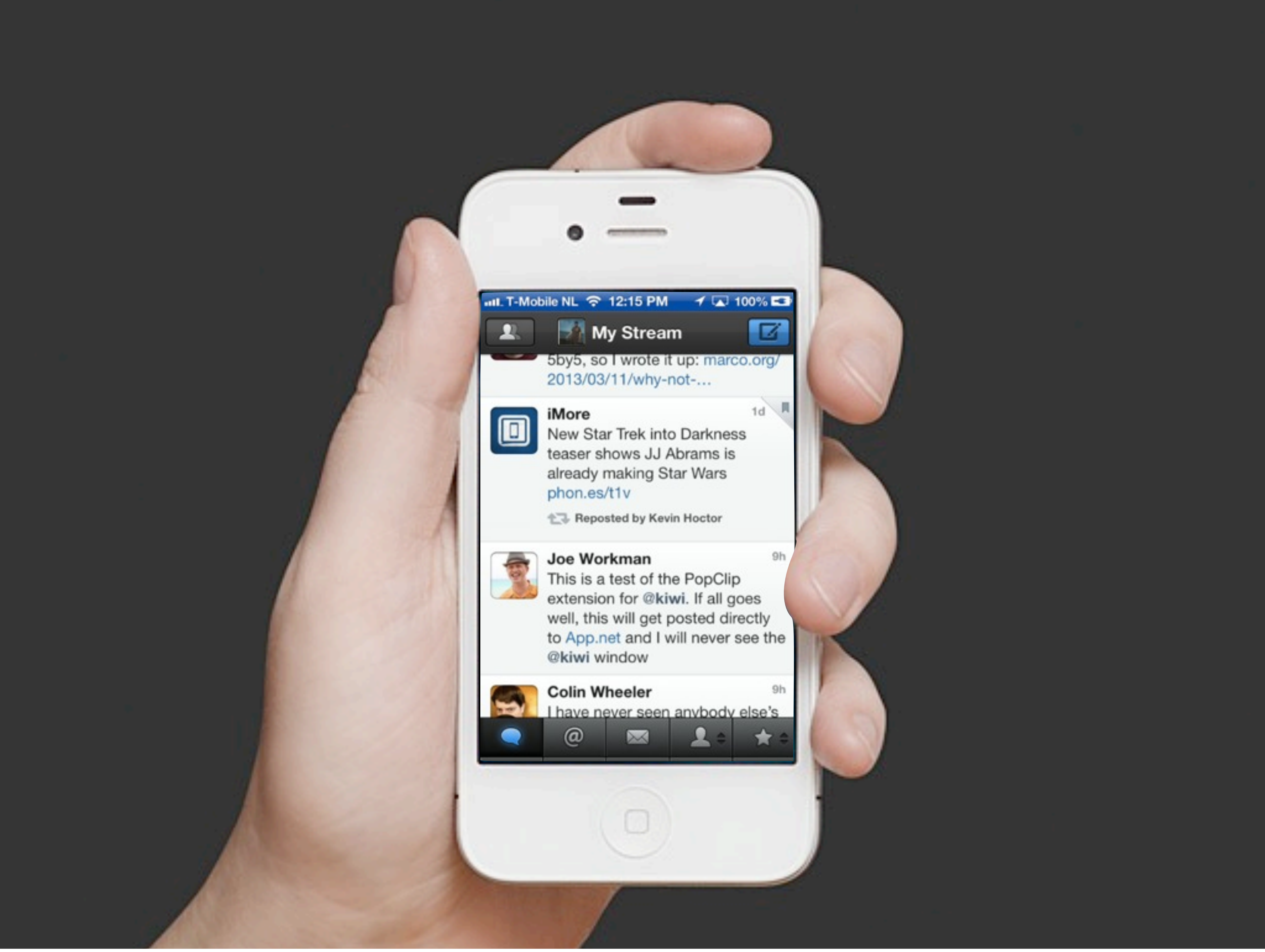


Wednesday, June 19, 13

Q: What goes in a viewcontroller?
A: All the things needed to present data into the view

Rule of Thumb: If it involves networking code, database code, drawing code, algorithms, it probably doesn't go in here.

The view controller should connect pieces, but not contain the logic for all those things...like a network stack, or a database...



5by5, so I wrote it up: marco.org/2013/03/11/why-not-…

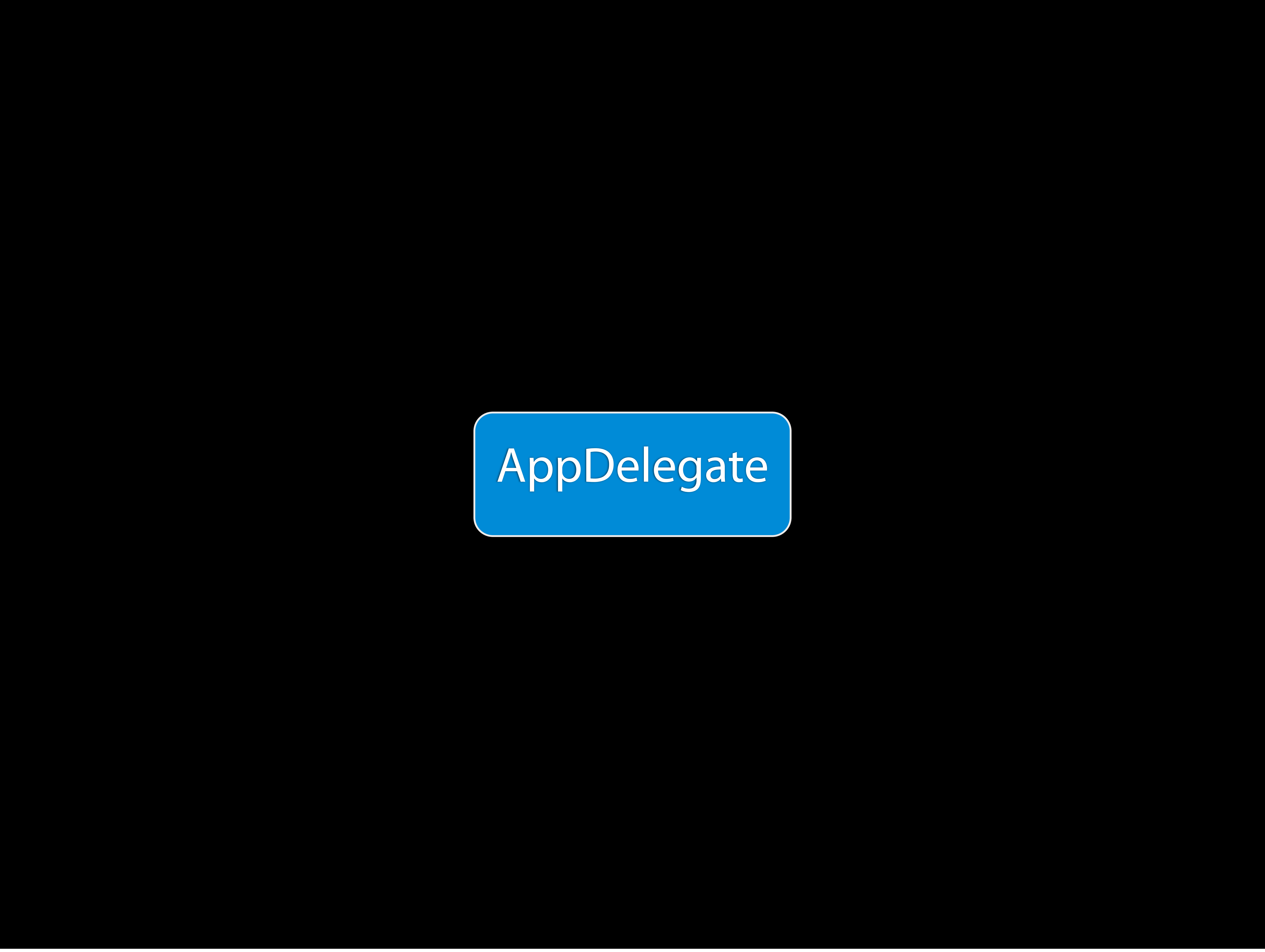
iMore 1d
New Star Trek into Darkness teaser shows JJ Abrams is already making Star Wars phon.es/t1v
Reposted by Kevin Hoctor

Joe Workman 9h
This is a test of the PopClip extension for @kiwi. If all goes well, this will get posted directly to App.net and I will never see the @kiwi window

Colin Wheeler 9h
I have never seen anybody else's

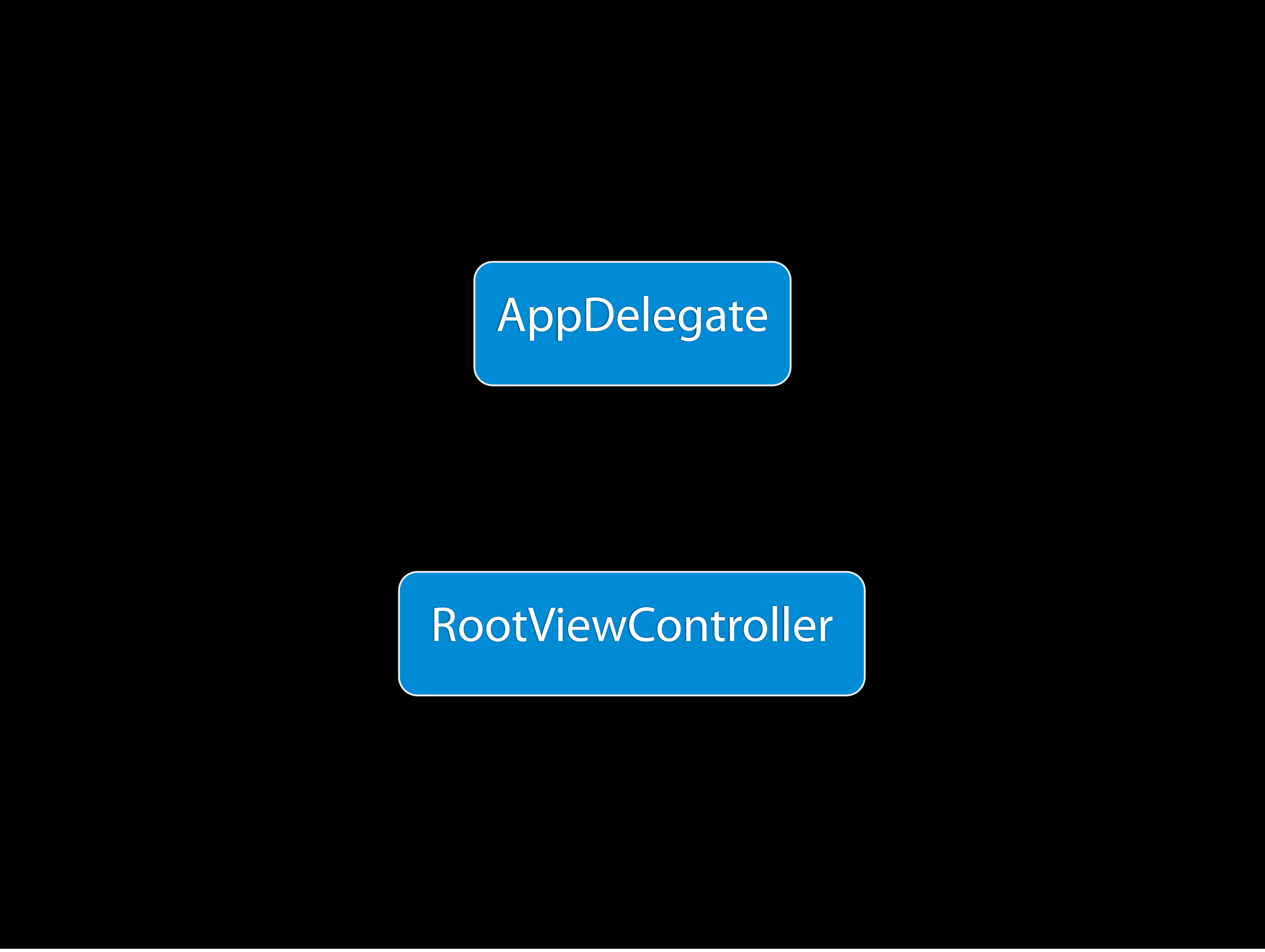
Wednesday, June 19, 13

Let’s get back to talking about our new mobile app



Data Flow Pattern
How does data flow in our app?

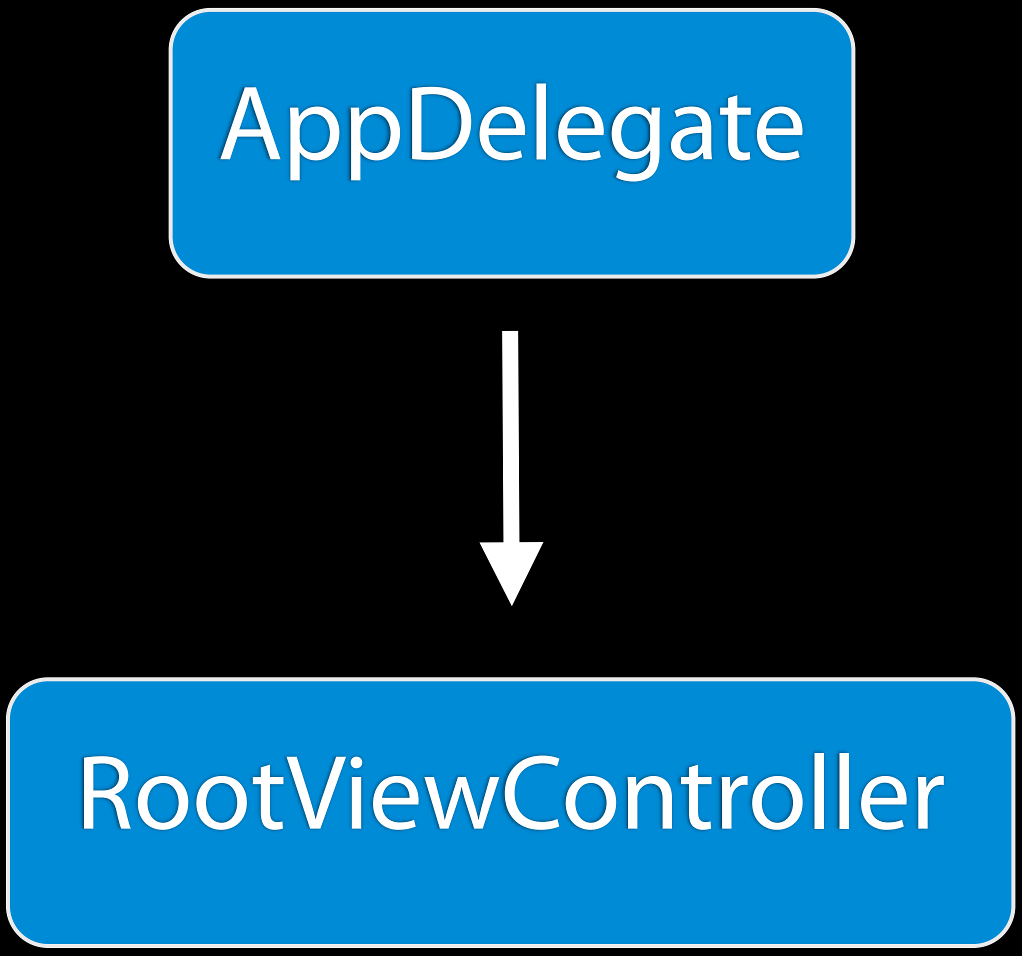
When you first start building an iOS App, you start with an AppDelegate



Wednesday, June 19, 13

Data Flow Pattern

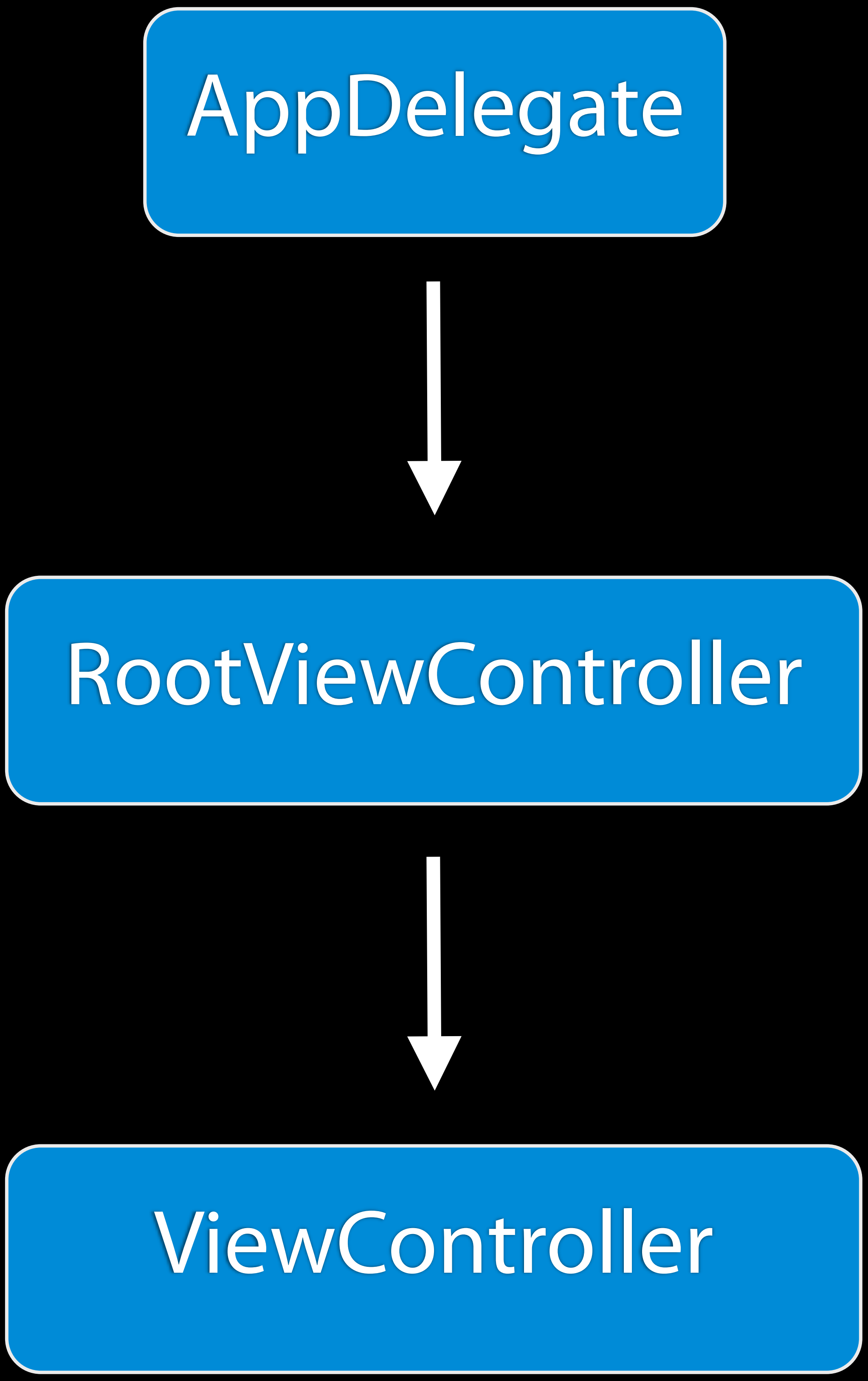
Then you add a root view controller because you can’t do anything in iOS without at least one view controller. This is added to the default window. Things are simple here



Wednesday, June 19, 13

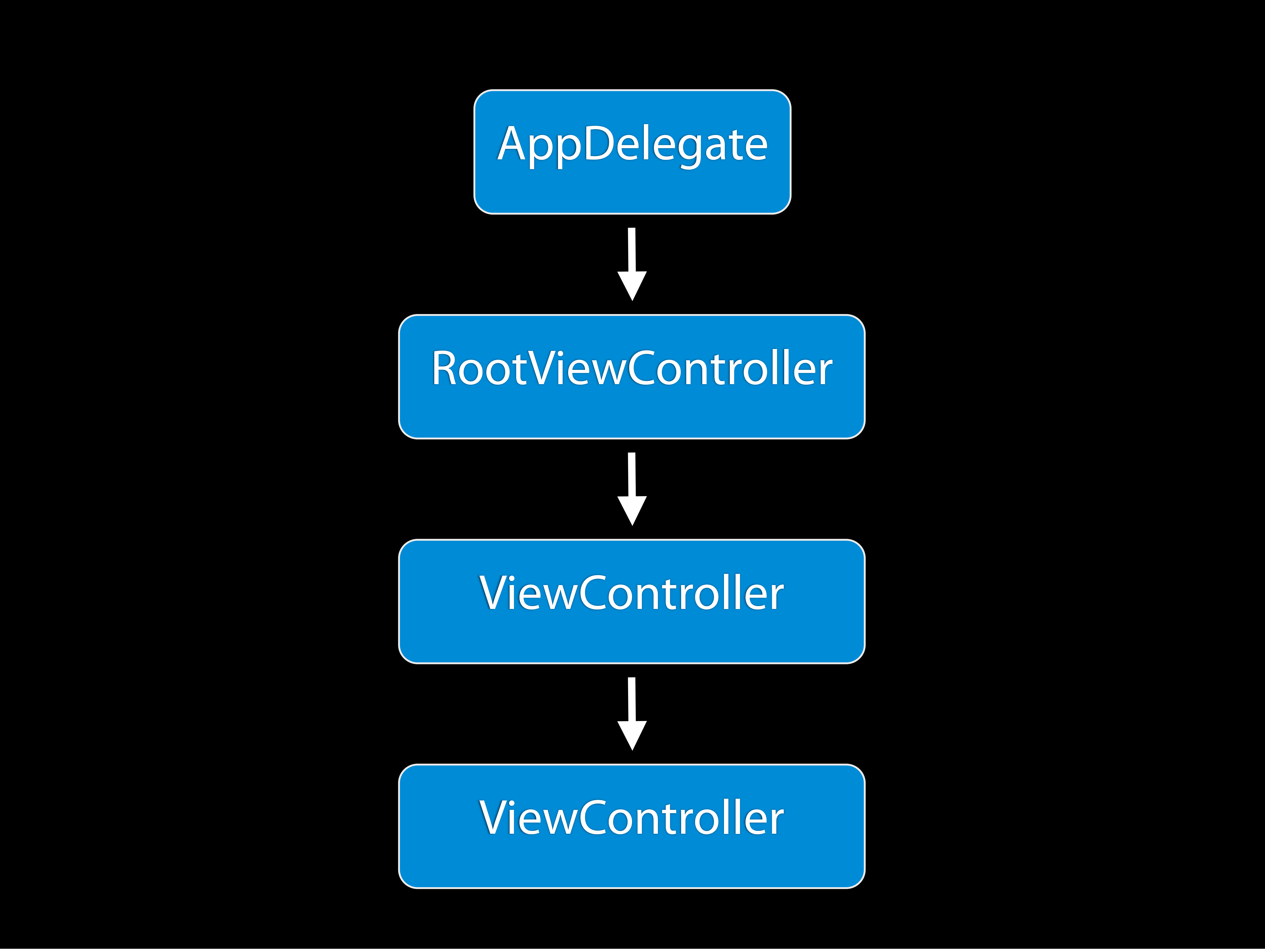
Data Flow Pattern

Then you add a root view controller because you can’t do anything in iOS without at least one view controller. This is added to the default window. Things are simple here

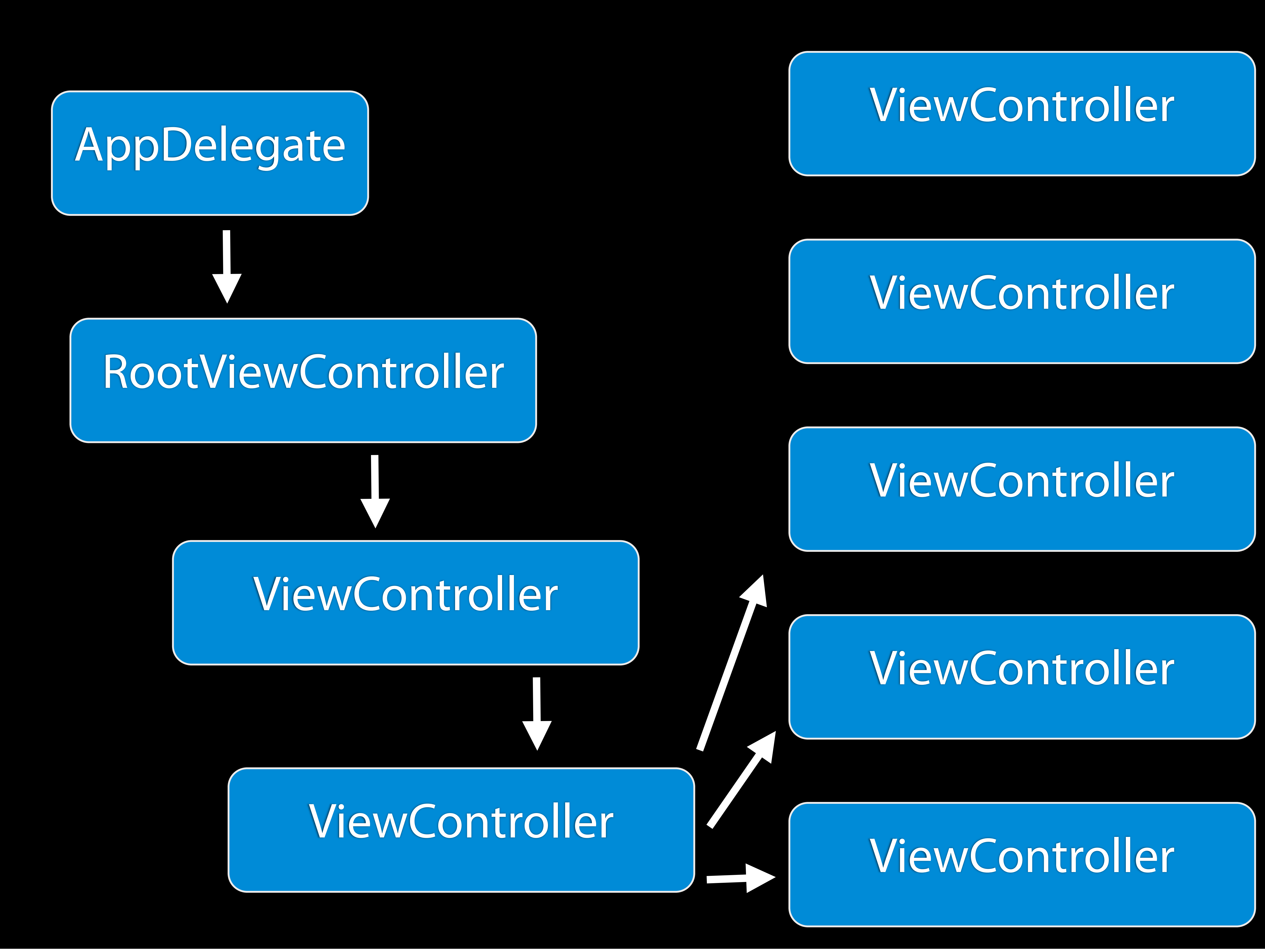


Wednesday, June 19, 13

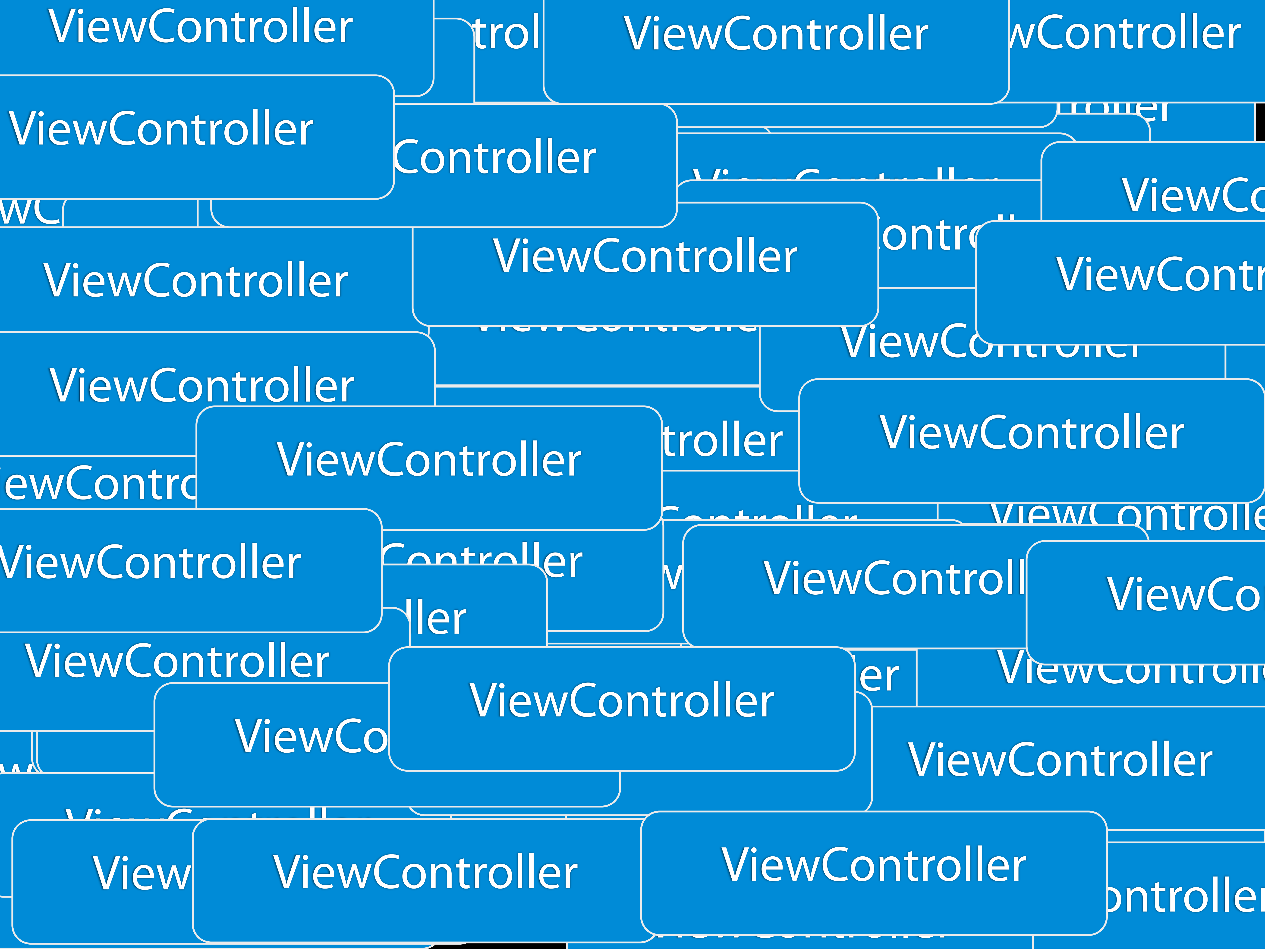
But then, you realize you need to add more viewcontrollers to the app



And more viewControllers...

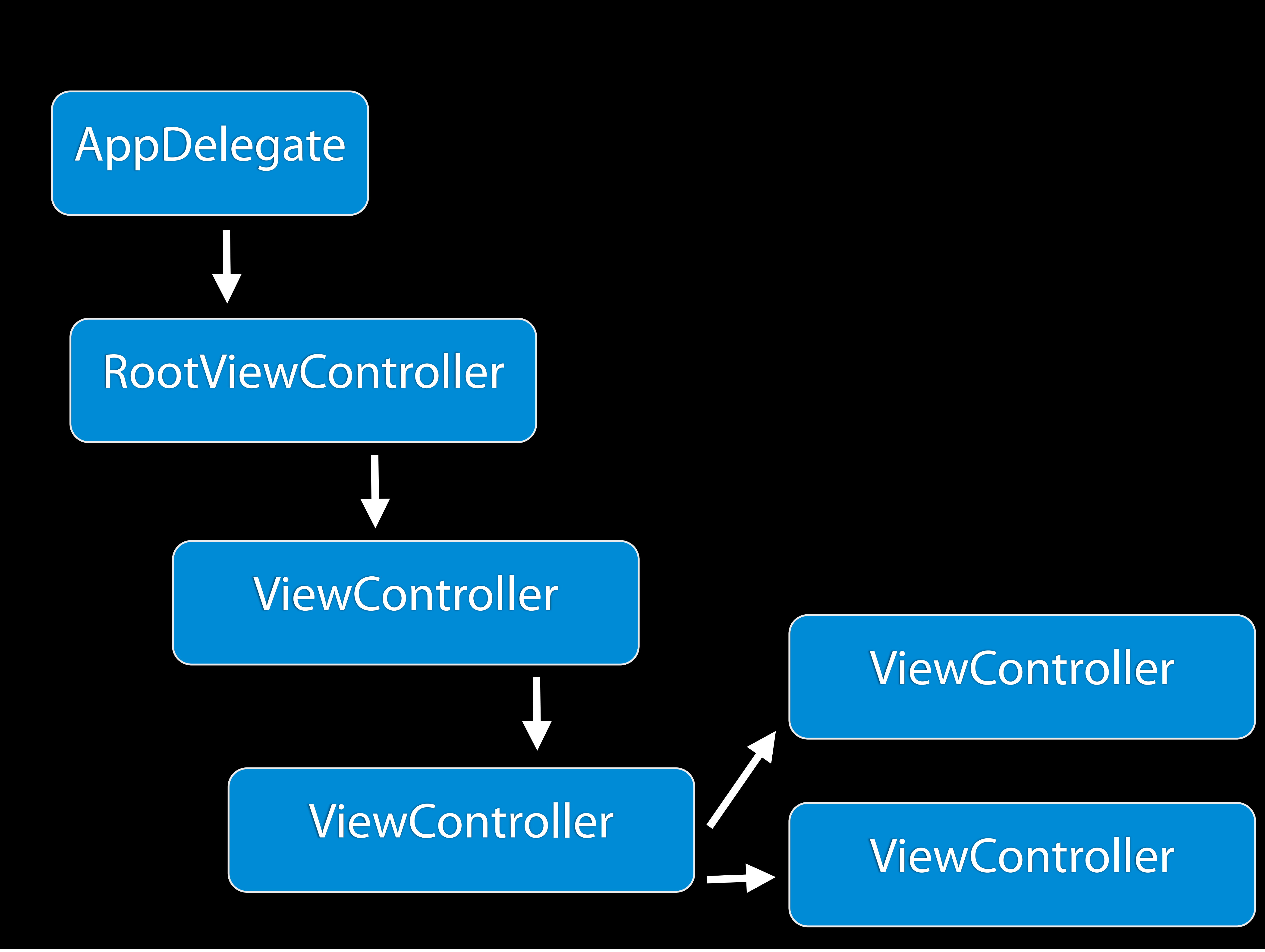


Until you get something like this, a hierarchy of view controllers.
Now, let me ask you, were does the networking component of this app go?

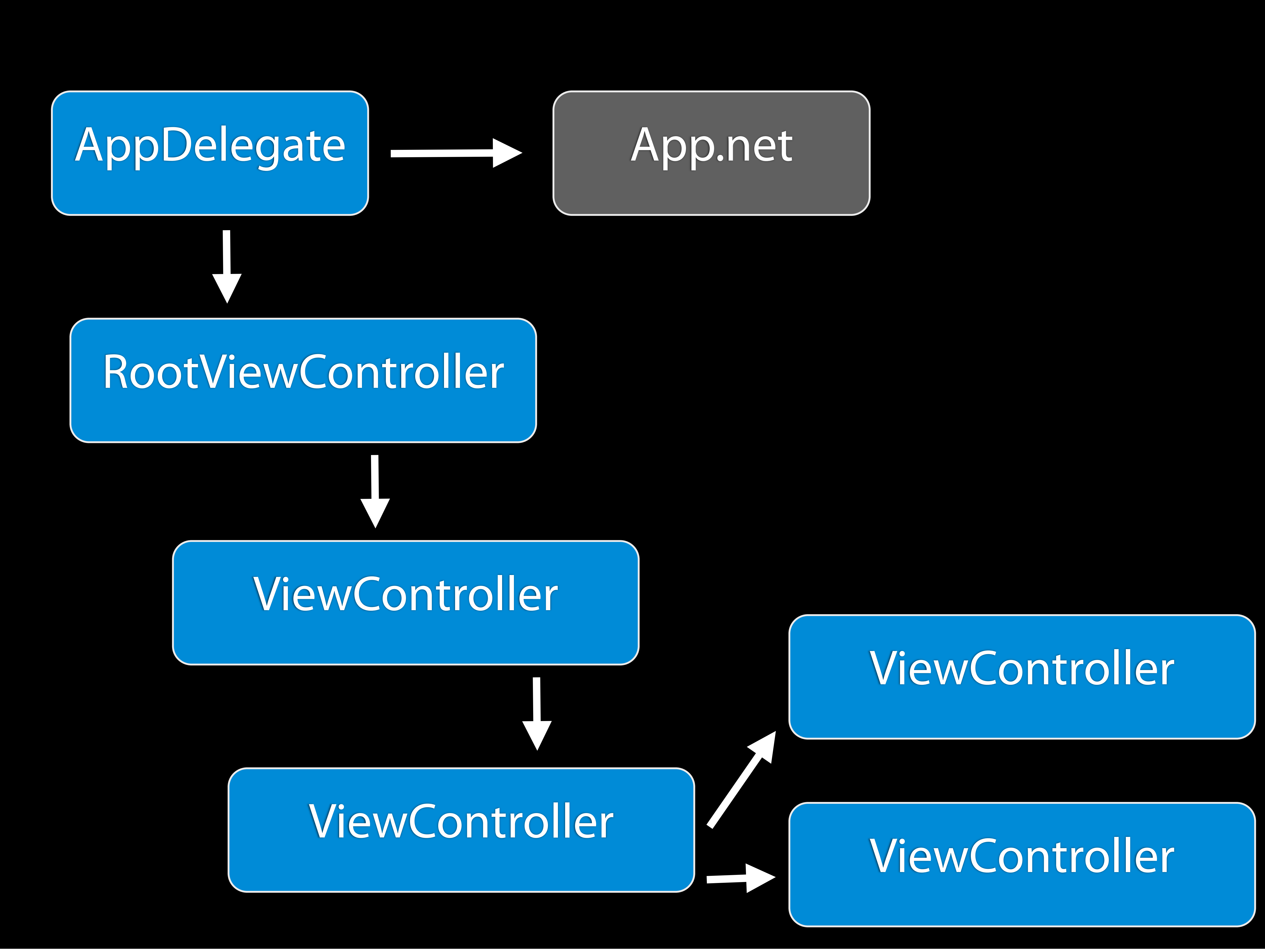


Wednesday, June 19, 13

Until you get something like this, a hierarchy of view controllers.
Now, let me ask you, were does the networking component of this app go?



Until you get something like this, a hierarchy of view controllers.
Now, were does the networking component of this app go?



Wednesday, June 19, 13

I propose that the app.net service attaches as a property on your app delegate for a couple reasons:
- the service is core to your app's functionality, put it at the core of your system
- the app delegate lives for the life of your app, so you can be assured this service is around and you have control over its lifecycle

You all probably do this already...I don't think it's wrong in principal after working with so many app architectures over the years. But I bet you all write the following line of code...

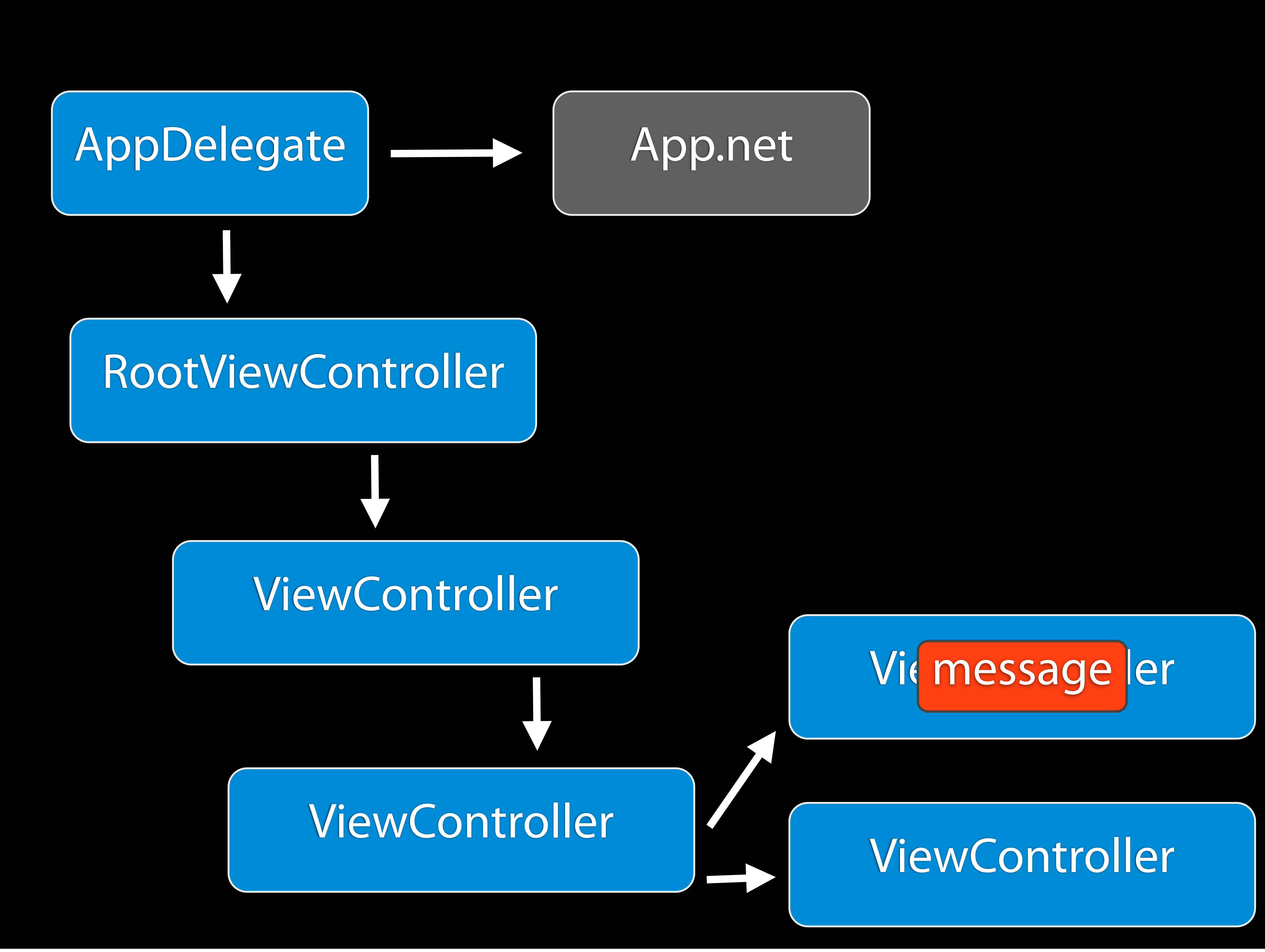
```
(MyAppDelegate *)[[UIApplication
sharedApplication] delegate]
```

But I’ve already seen this before!

```
#define sharedDelegate \
(MyAppDelegate *)[[UIApplication sharedApplication] delegate]
```

Wednesday, June 19, 13

And then some of you do this! PLEASE STOP DOING THIS!



I propose that the app.net service attaches as a property on your app delegate for a couple reasons:
- the service is core to your app’s functionality, put it at the core of your system

You all probably do this already...I don’t think it’s wrong in principal after working with so many app architectures over the years. But I bet you all write the following line of code...

Pause here more, explain more :

What
When to use
Why

Chain of Responsibility

http://en.wikipedia.org/wiki/Chain-of-responsibility_pattern

Some properties of the chain of responsibility

Responder Chain

Wednesday, June 19, 13

In Cocoa, we have the responder chain

```
[[UIApplication
sharedApplication]
sendAction:to:from:forEvent:]
```

Now, this may look like “more code” but it is so much more powerful and flexible. This code uses the application singleton, which knows about the view hierarchy since that’s how events are sent to your viewcontrollers, to send your action up the chain.

```
@implementation UIView
(FindAndResignFirstResponder)

- (BOOL)findAndResignFirstResponder
{
    if ([self isFirstResponder]) {
        [self resignFirstResponder];
        return YES;
    }
    for (UIView *subView in [self subviews]) {
        if ([subView findAndResignFirstResponder])
            return YES;
    }
    return NO;
}
@end
```

http://stackoverflow.com/questions/1823317/get-the-current-first-responder-without-using-a-private-api

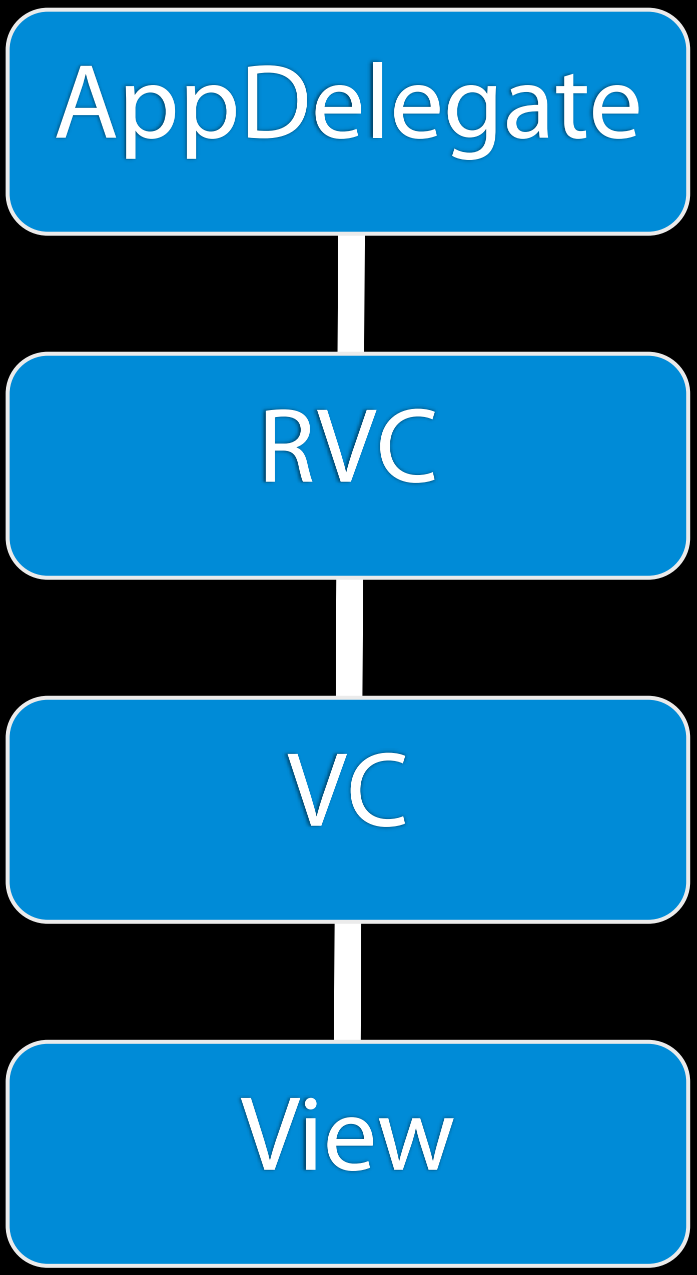
No doubt, many of you have helper methods like this.

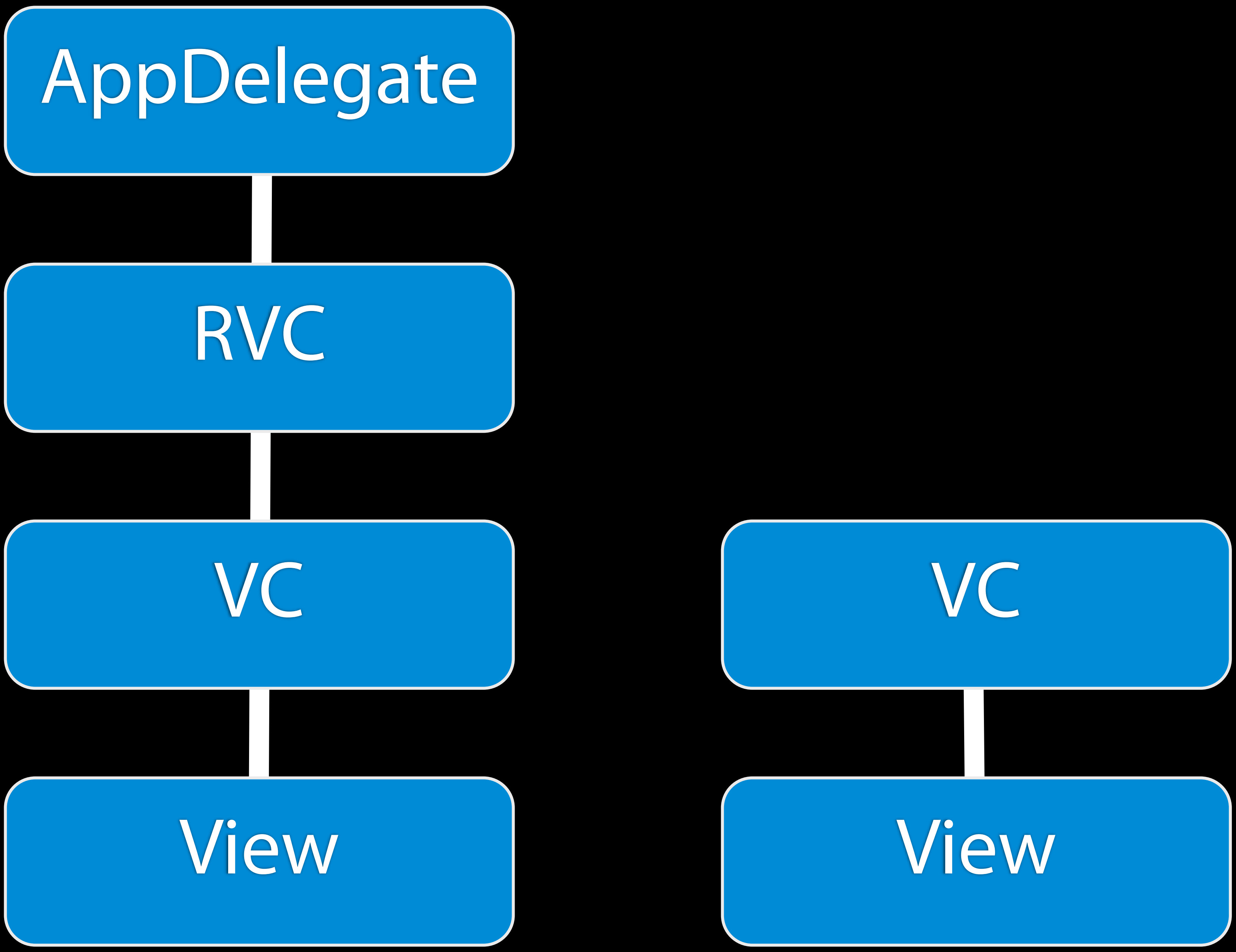
```
[[UIApplication sharedApplication]
         sendAction:@selector(resignFirstResponder)
                to:nil
              from:self
          forEvent:nil]
```

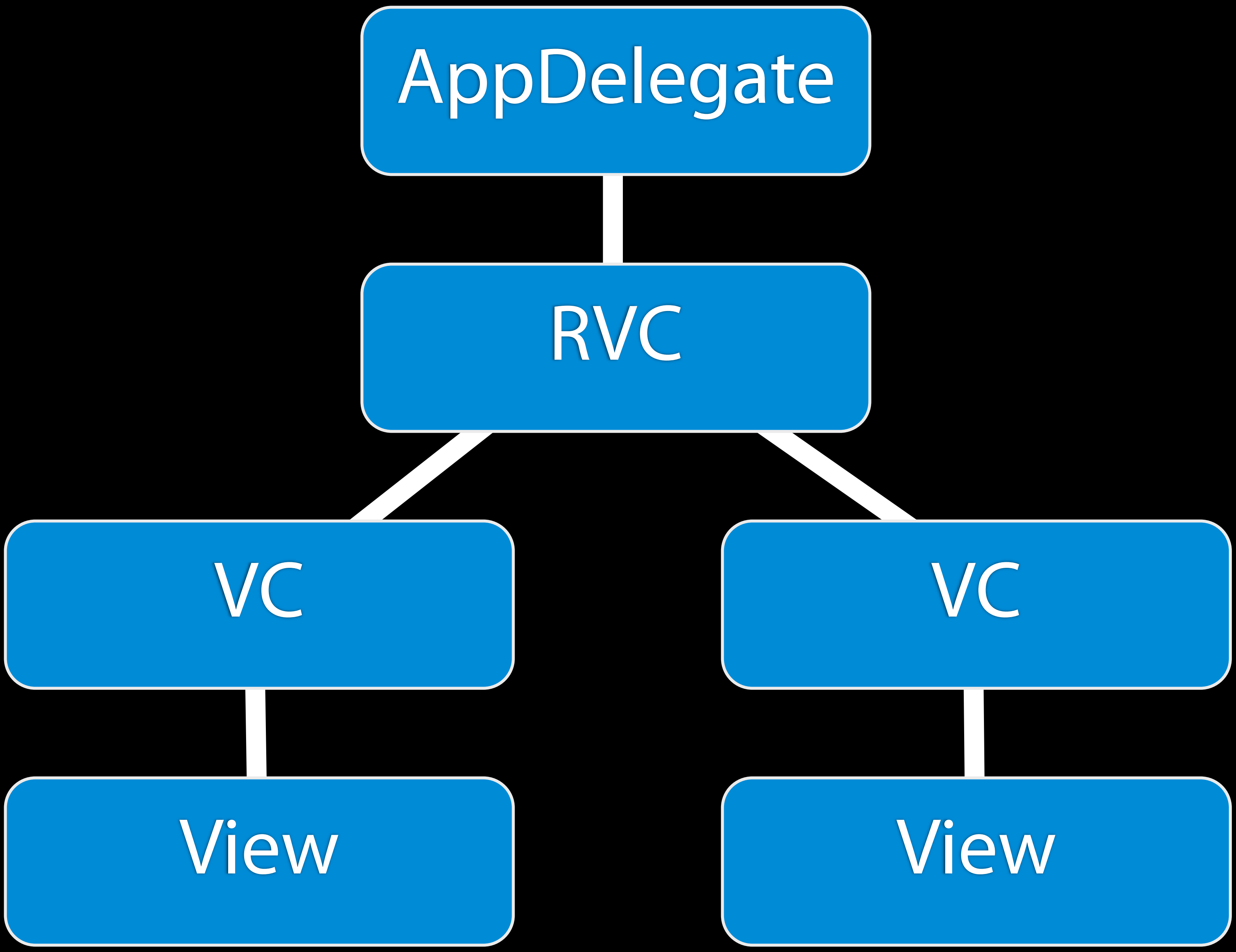
Also note the from: parameter here. 'self' is the object in the responder chain. This parameter (from:) cannot be for this to work. The responder chain is a way to respond to events. If no one sent the action, what is there to respond to?

- (void)viewDidAppear:

This technique, because it’s part of the Cocoa view hierarchy, can only work when your view controller is attached to the hierarchy. The best time is on viewDidAppear in your view controllers.


AppDelegate
RVC
VC
View


AppDelegate
RVC
VC
View
VC
View


AppDelegate
RVC
VC
VC
View
View

Commands and network service patterns

AFNetworking

Network Commands

```
- (void)viewDidLoad {
    [super viewDidLoad];

    // Setting Up Table View
    self.tableView = [[UITableView alloc] initWithFrame:CGRectMake(0.0, 0.0, self.view.bounds.size.width,
self.view.bounds.size.height) style:UITableViewStylePlain];
    self.tableView.dataSource = self;
    self.tableView.delegate = self;
    self.tableView.autoresizingMask = UIViewAutoresizingFlexibleWidth | UIViewAutoresizingFlexibleHeight;
    self.tableView.hidden = YES;
    [self.view addSubview:self.tableView];

    // Setting Up Activity Indicator View
    self.activityIndicatorView = [[UIActivityIndicatorView alloc]
initWithActivityIndicatorStyle:UIActivityIndicatorViewStyleGray];
    self.activityIndicatorView.hidesWhenStopped = YES;
    self.activityIndicatorView.center = self.view.center;
    [self.view addSubview:self.activityIndicatorView];
    [self.activityIndicatorView startAnimating];

    // Initializing Data Source
    self.movies = [[NSArray alloc] init];

    NSURL *url = [[NSURL alloc] initWithString:@"http://itunes.apple.com/search?term=harry&country=us&entity=movie"];
    NSURLRequest *request = [[NSURLRequest alloc] initWithURL:url];

    AFJSONRequestOperation *operation = [AFJSONRequestOperation JSONRequestOperationWithRequest:request
success:^(NSURLRequest *request, NSHTTPURLResponse *response, id JSON) {
        self.movies = [JSON objectForKey:@"results"];
        [self.activityIndicatorView stopAnimating];
        [self.tableView setHidden:NO];
        [self.tableView reloadData];

    } failure:^(NSURLRequest *request, NSHTTPURLResponse *response, NSError *error, id JSON) {
        NSLog(@"Request Failed with Error: %@, %@", error, error.userInfo);
    }];

    [operation start];
}
```

Zoom and enhance

```
    // Initializing Data Source
    self.movies = [[NSArray alloc] init];

    NSURL *url = [[NSURL alloc] initWithString:@"http://itunes.apple.com/search?
term=harry&country=us&entity=movie"];
    NSURLRequest *request = [[NSURLRequest alloc] initWithURL:url];

    AFJSONRequestOperation *operation = [AFJSONRequestOperation
JSONRequestOperationWithRequest:request success:^(NSURLRequest *request,
NSHTTPURLResponse *response, id JSON) {
        self.movies = [JSON objectForKey:@"results"];
        [self.activityIndicatorView stopAnimating];
        [self.tableView setHidden:NO];
        [self.tableView reloadData];

    } failure:^(NSURLRequest *request, NSHTTPURLResponse *response, NSError
*error, id JSON) {
        NSLog(@"Request Failed with Error: %@, %@", error, error.userInfo);
    }];

    [operation start];
```

Wednesday, June 19, 13

Zoom and enhance

```
^(NSURLRequest *request, NSHTTPURLResponse *response, id JSON) {
        self.movies = [JSON objectForKey:@"results"];
        [self.activityIndicatorView stopAnimating];
        [self.tableView setHidden:NO];
        [self.tableView reloadData];
    }
```

Wednesday, June 19, 13

This return block has elements from all three areas combined
Self.movies lives in the view controller
And the block is a result of a network stack

ADN Service

Wednesday, June 19, 13

ADN Service

https://alpha-api.app.net/stream/0/posts/stream

ADN Service

https://alpha-api.app.net/stream/0/posts/stream

POST:
include_reposters=1&include_annotations=1&include_muted=0&include_starred_by=1

ADN Service

Get Personal Stream

ADN Service

Get Personal Stream

include_reposters

include_muted

include_annotations

include_starred_by

ADN Service

Get Personal Stream

```
@property (nonatomic, assign) BOOL includeReposters

@property (nonatomic, assign) BOOL includeMuted

@property (nonatomic, assign) BOOL includeAnnotations

@property (nonatomic, assign) BOOL includeStarredBy
```

ADN Service

```
@class ADNRetrievePersonalStreamCommand : ADNCommand

    @property (nonatomic, assign) BOOL includeReposters
    @property (nonatomic, assign) BOOL includeMuted
    @property (nonatomic, assign) BOOL includeAnnotations
    @property (nonatomic, assign) BOOL includeStarredBy
```

```
@class ADNService : UIResponder
    @property (nonatomic, copy) NSURL *baseURL;
```

ADN Service

```
@class ADNRetrievePersonalStreamCommand : ADNCommand

    @property (nonatomic, assign) BOOL includeReposters
    @property (nonatomic, assign) BOOL includeMuted
    @property (nonatomic, assign) BOOL includeAnnotations
    @property (nonatomic, assign) BOOL includeStarredBy
```

ADN Service

Get Personal Stream

Command

ADNCommand

ADNCommand

ADNPersonalStreamCommand

$$\begin{tikzcd}\text{ADNCommand} \\ \text{ADNPersonalStreamCommand} \arrow[u]\end{tikzcd}$$

ADNCommand

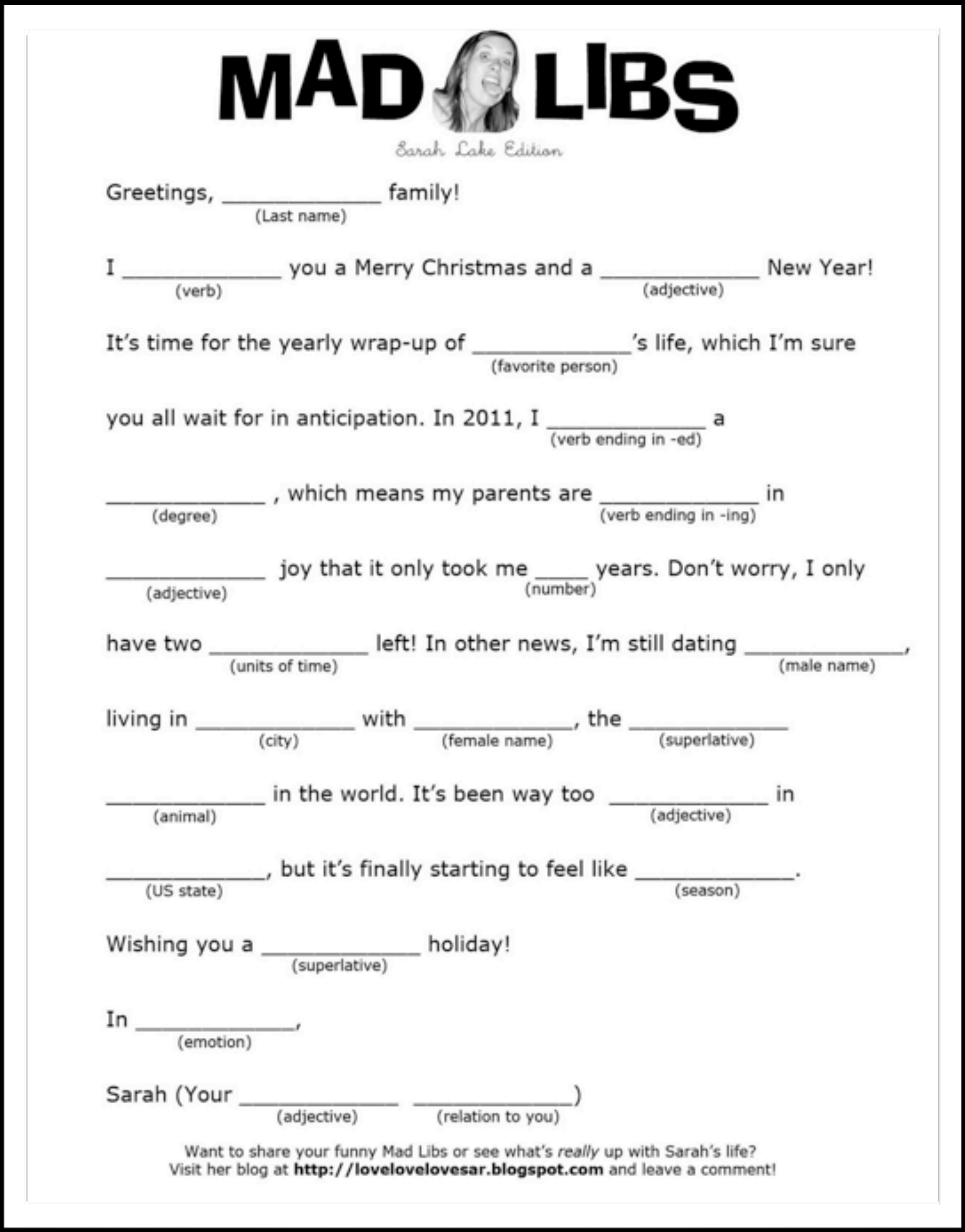
ADNPersonalStreamCommand

ADNPostStatusCommand

ADNRetrieveChannelCommand

ADNUserLookupCommand

Template



MAD LIBS

Sarah Lake Edition

Greetings, _____________ family!
(Last name)

I _____________ you a Merry Christmas and a _____________ New Year!
(verb) (adjective)

It's time for the yearly wrap-up of _____________'s life, which I'm sure
(favorite person)

you all wait for in anticipation. In 2011, I _____________ a
(verb ending in -ed)

_____________ , which means my parents are _____________ in
(degree) (verb ending in -ing)

_____________ joy that it only took me ____ years. Don't worry, I only
(adjective) (number)

have two _____________ left! In other news, I'm still dating _____________,
(units of time) (male name)

living in _____________ with _____________, the _____________
(city) (female name) (superlative)

_____________ in the world. It's been way too _____________ in
(animal) (adjective)

_____________, but it's finally starting to feel like _____________.
(US state) (season)

Wishing you a _____________ holiday!
(superlative)

In _____________,
(emotion)

Sarah (Your _____________ _____________)
(adjective) (relation to you)

Want to share your funny Mad Libs or see what's *really* up with Sarah's life?
Visit her blog at **http://lovelovelovesar.blogspot.com** and leave a comment!

```
//AppDelegate.h
@interface AppDelegate : UIResponder<UIApplicationDelegate>

@property (nonatomic, strong, readwrite) IBOutlet UIWindow *window;
@property (nonatomic, strong, readonly) WebService *webService;

@end




//AppDelegate.m
- (UIResponder *)nextResponder;
{
    return self.webService;
}
```

```
@interface WebService : UIResponder

@property (nonatomic, copy, readonly) NSURL *baseURL;
@property (nonatomic, strong, readonly) TokenStorage *tokenStorage;
//..more properties

- (id) initWithBaseURL:(NSURL *)baseURL;
- (void) sendCommand:(WebServiceCommand *)command;

- (void) presentMessageFromCommand:(NSString *)message;
- (BOOL) isAuthenticated;

- (BOOL) performAction:(SEL)action from:(id)sender;
- (BOOL) performAction:(SEL)action from:(id)sender parameters:
(NSDictionary *)parameters;

+ (BOOL) isNetworkReachable;

@end
```

```
@interface WebService (Commands)

- (IBAction) loginToService;
- (IBAction) logoutFromService;

- (IBAction) retrieveUsersPersonalStream:(id)sender;

@end
```

The Web Service Commands category

```
- (void) loginToService
{
    WebServiceLoginCommand *command = [WebServiceLoginCommand
commandWithService:self];
    command.tokenStorage = self.tokenStorage;
    command.username = [self.delegate username];
    command.password = [self.delegate password];
    [command send];
}
```

Wednesday, June 19, 13

In the Web Service category

```
- (NSString *)username;
{
    return [self textValueForCellAtIndexPath:[NSIndexPath
indexPathForRow:0 inSection:0]];
}

- (NSString *)password;
{
    return [[self textValueForCellAtIndexPath:[NSIndexPath
indexPathForRow:1 inSection:0]] lowercaseString];
}

- (IBAction) login:(id)sender;
{
    if ([[self username] length] && [[self password] length])
    {
        self.messageLabel.text = @"";
        [[UIApplication sharedApplication]
performActionInResponderChain:@selector(resignFirstResponder)
from:self];
        [SVProgressHUD showWithStatus:@"Logging in..."
maskType:SVProgressHUDMaskTypeBlack];
    }
}
```

In the Login View

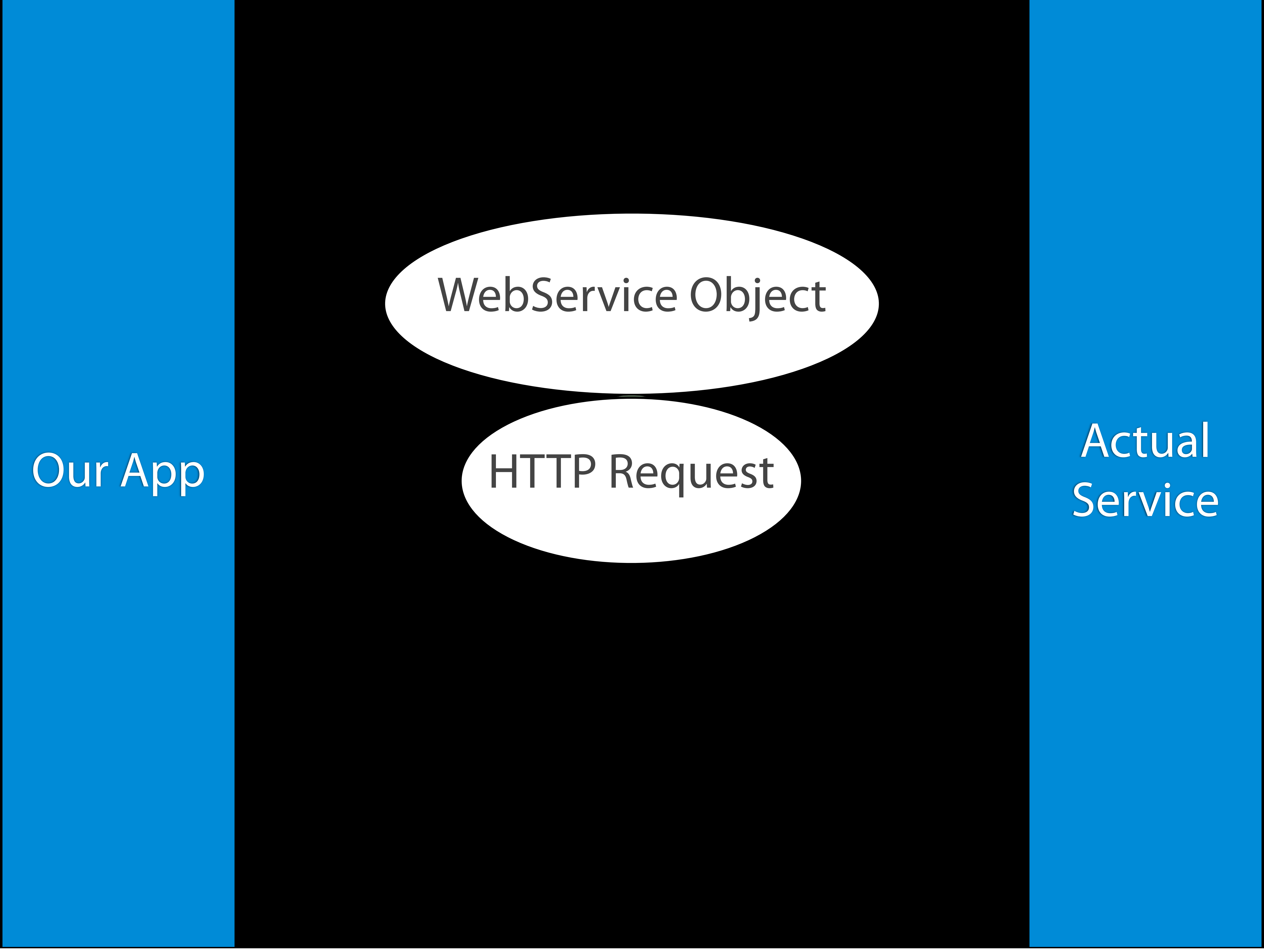
```
- (void) sendCommand:(WebServiceCommand *)command;
{
    BOOL commandVerified = [self verifyCommand:command];
    if (!commandVerified) return;

    if ([self.httpClient networkReachabilityStatus] ==
AFNetworkReachabilityStatusNotReachable)
    {
        [self.delegate displayMessage:@"No Internet Connection"];
        DDLogInfo(@"Not connected to a network");
    }
    else
    {
        AFHTTPRequestOperation *operation = [command createRequestOperation];

        [self.httpClient enqueueHTTPRequestOperation:operation];

        DDLogInfo(@"Sent Command: %@", command);
    }
}
```

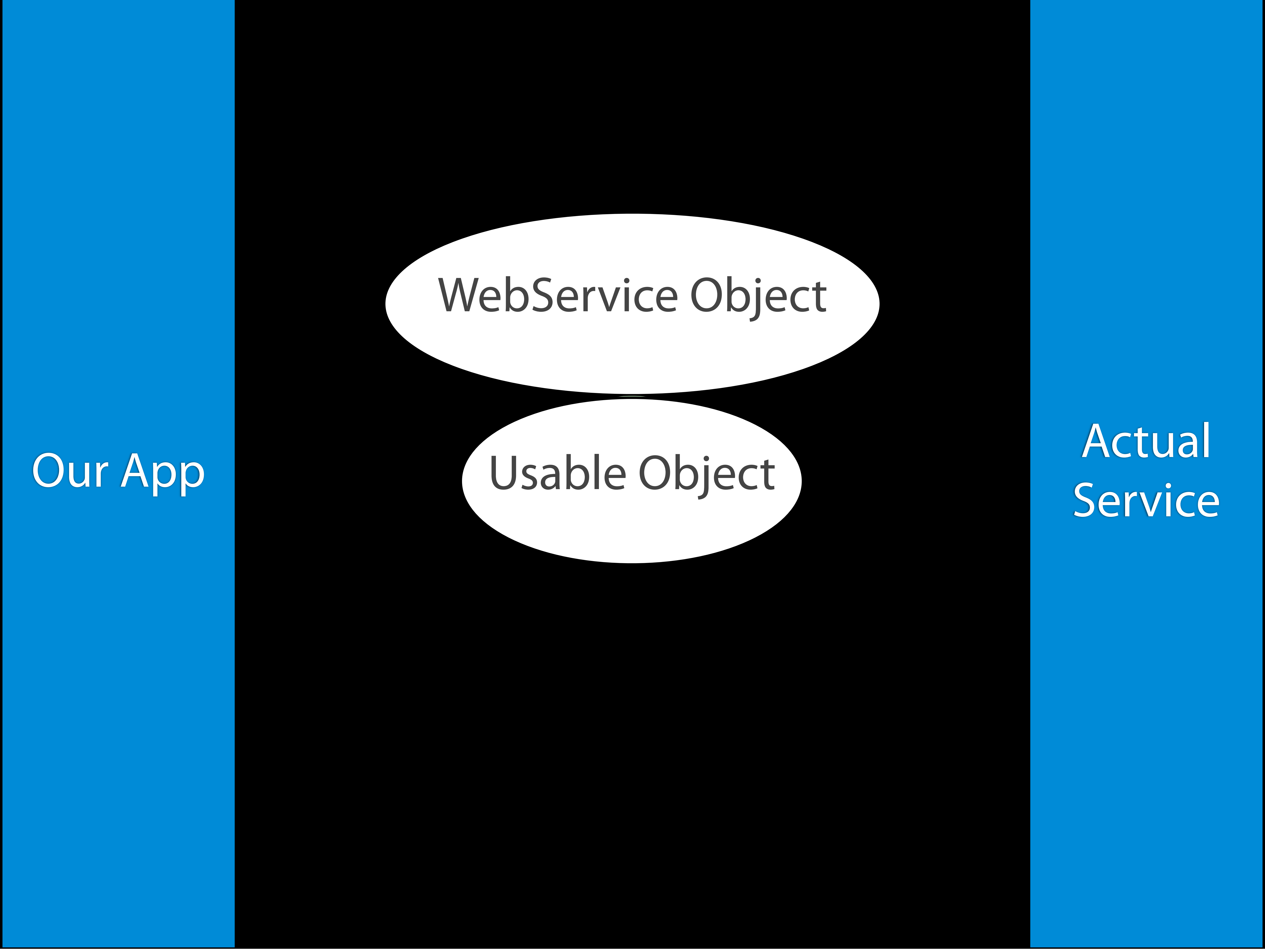
In the Web Service sendCommand method

Lets take a look at the two statements afterwards

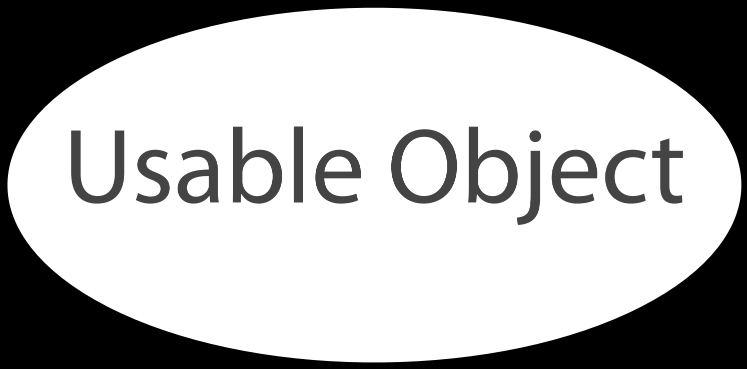
```
[self.delegate displayMessage:@"No Internet Connection"];
DDLogInfo(@"Not connected to a network");
```

```
AFHTTPRequestOperation *operation = [command
createRequestOperation];

[self.httpClient enqueueHTTPRequestOperation:operation];

DDLogInfo(@"Sent Command: %@", command);
```


Our App
WebService Object
HTTP Request
Actual Service


Our App
WebService Object
Usable Object
Actual Service

Needs to be more clear



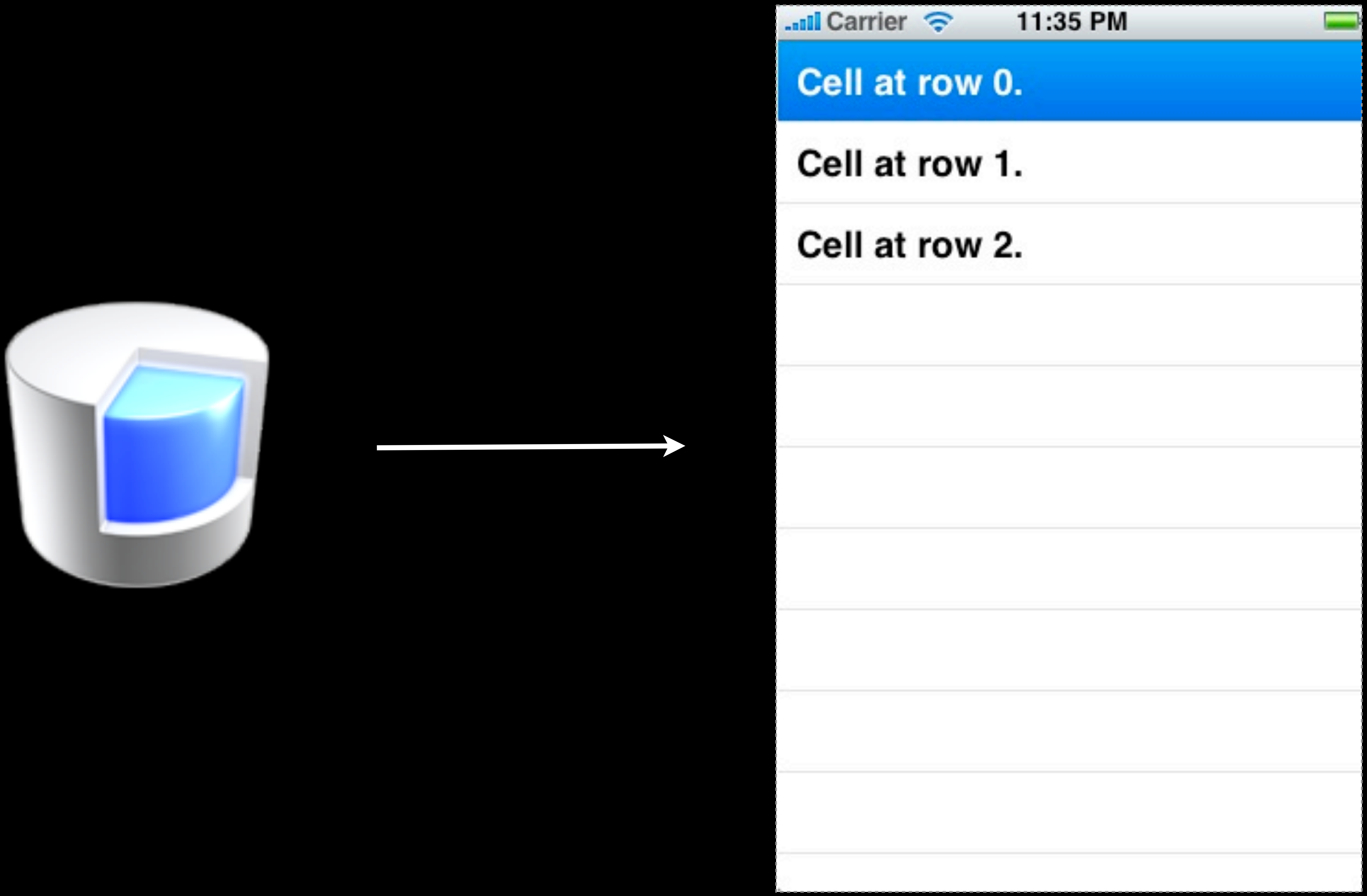
Now that we have a UI, and network support, we need to store and display that data.
The AppDelegate also has a reference to the data store, but it’s merely another responder in the chain, and is also an injected dependency into the webservice

Usable Object

Needs to be more clear

Wednesday, June 19, 13

Now that we have a UI, and network support, we need to store and display that data.
The AppDelegate also has a reference to the data store, but it's merely another responder in the chain, and is also an injected dependency into the webservice



Wednesday, June 19, 13

Now that we have a UI, and network support, we need to store and display that data.

NSFetchedResultsController

```
- (void) loadUserStream;
{
    self.results = [StreamEvent MR_fetchAllSortedBy:StreamEvent.createdDate
                                          ascending:YES
                                      withPredicate:[self streamFilter]
                                            groupBy:nil
                                           delegate:self
                                          inContext:self.context];
}
```

Wednesday, June 19, 13

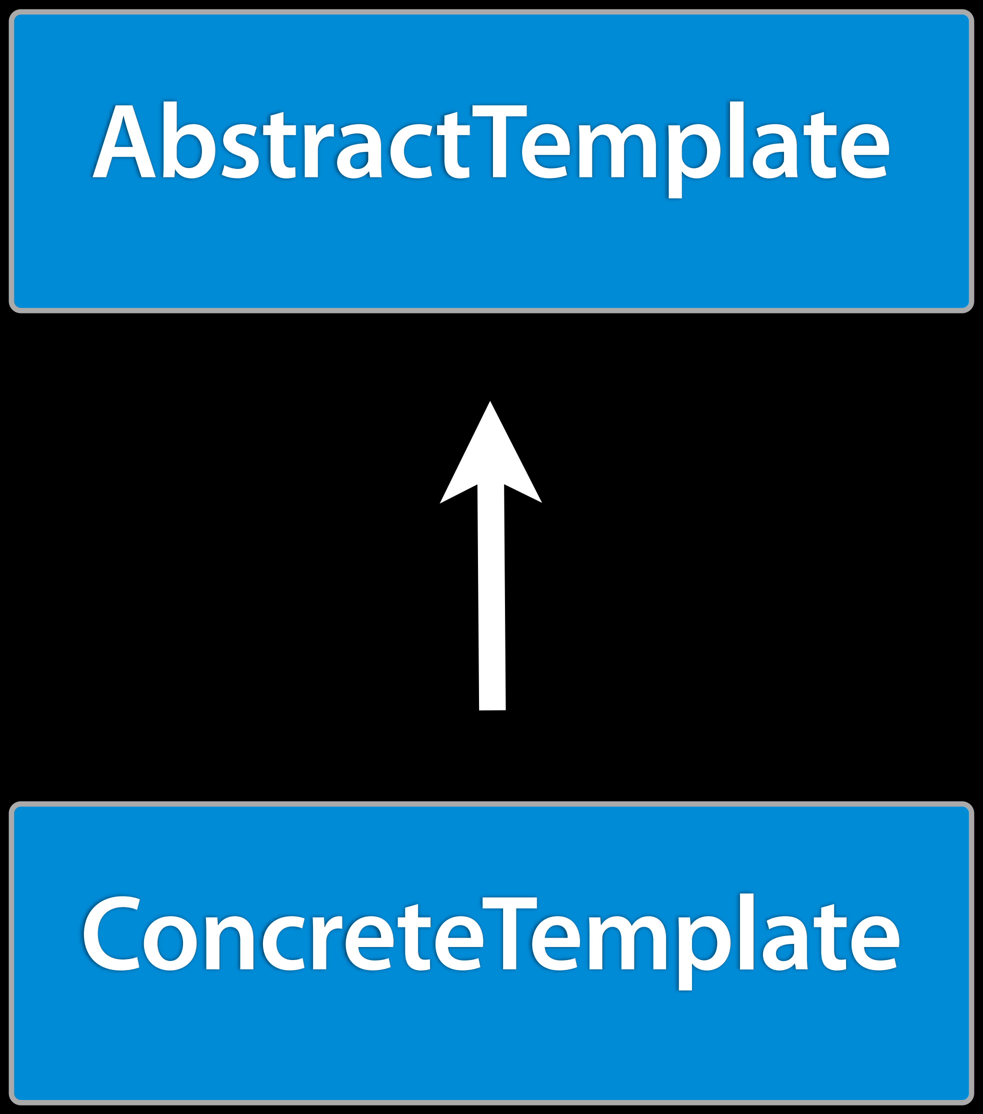
NSFetchedResultsControllerDelegate

Delegation

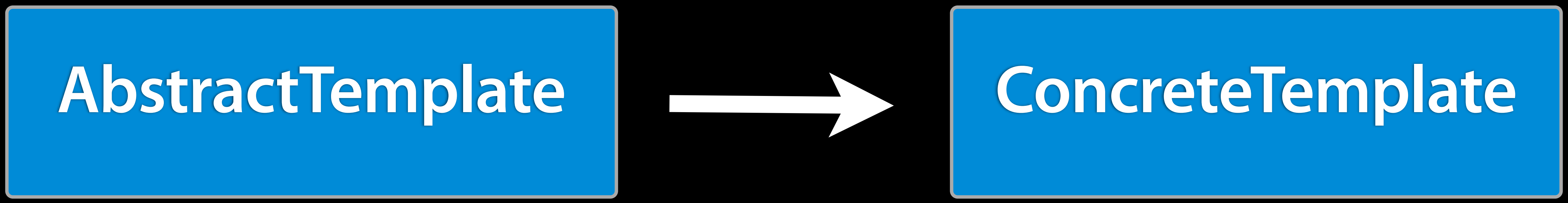
Delegate pattern is like the template pattern, but with instances rather than classes

https://developer.apple.com/library/mac/#documentation/General/Conceptual/DevPedia-CocoaCore/Delegation.html

This tells the view controller what to do, but now what about the view?



Delegation is like the template method, you still fill in the blanks



Wednesday, June 19, 13

But when you delegate, your delegate object does not necessarily subclass from your template. I like to think of a delegate object as a side-by-side template, or peer object template. You have to manually implement some things you'd normally get for free in the object oriented way of subclassing, such as checking if your delegate implements a method beforehand. But this is great for frameworks, so you don't have to carry all the baggage that comes along with subclassing.

```
- (void)controllerWillChangeContent:(NSFetchedResultsController
*)controller
```

```
- (void)controller:(NSFetchedResultsController *)controller
   didChangeObject:(id)anObject
       atIndexPath:(NSIndexPath *)theIndexPath
     forChangeType:(NSFetchedResultsChangeType)type
      newIndexPath:(NSIndexPath *)newIndexPath
```

```
- (void)controllerDidChangeContent:(NSFetchedResultsController
*)controller
```

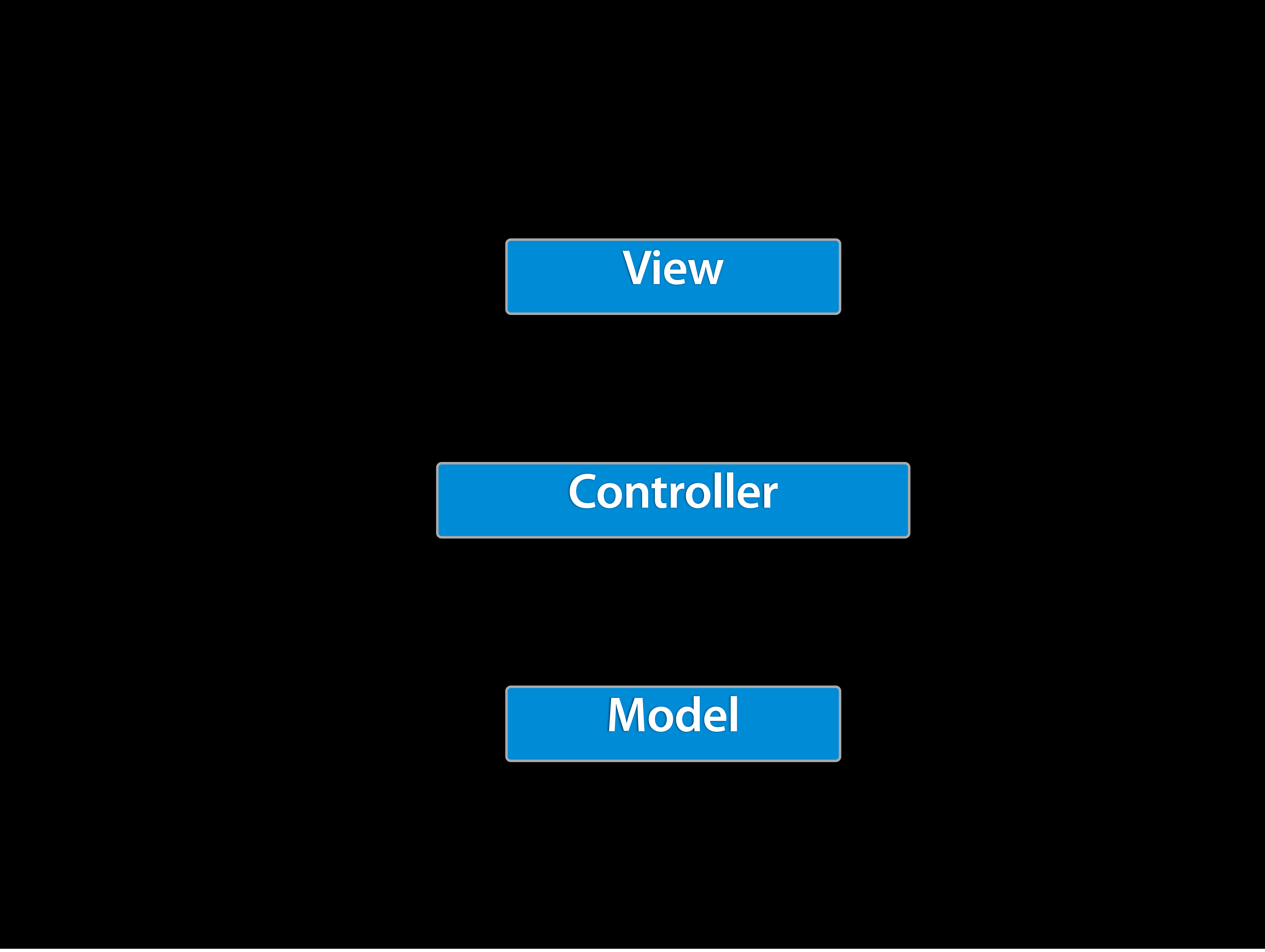
```objc
    switch(type)
    {
        case NSFetchedResultsChangeInsert:
            [tableView insertRowsAtIndexPaths:@[theIndexPath]
withRowAnimation:UITableViewRowAnimationNone];
            break;

        case NSFetchedResultsChangeDelete:
            [self verifyProductPredicate:anObject];
            [tableView deleteRowsAtIndexPaths:@[theIndexPath]
withRowAnimation:UITableViewRowAnimationFade];
            break;

        case NSFetchedResultsChangeUpdate:
        {
            id object = [self.results objectAtIndexPath:theIndexPath];
            [self configureCell:(id)[tableView
cellForRowAtIndexPath:theIndexPath] forObject:object];
        }
            break;

        case NSFetchedResultsChangeMove:
            [tableView deleteRowsAtIndexPaths:@[theIndexPath]
withRowAnimation:UITableViewRowAnimationFade];
            [tableView insertRowsAtIndexPaths:@[theIndexPath]
withRowAnimation:UITableViewRowAnimationFade];
            break;
    }
```

Last steps from controller to view with observer pattern

Observer

Wednesday, June 19, 13

The observer pattern is built into Objective C these days. In our example here, NSFRC is what handles the observation for us.



Wednesday, June 19, 13

We can now take a fresh look at what model-view-controller means after building an app that has taken advantage of patterns.

Command

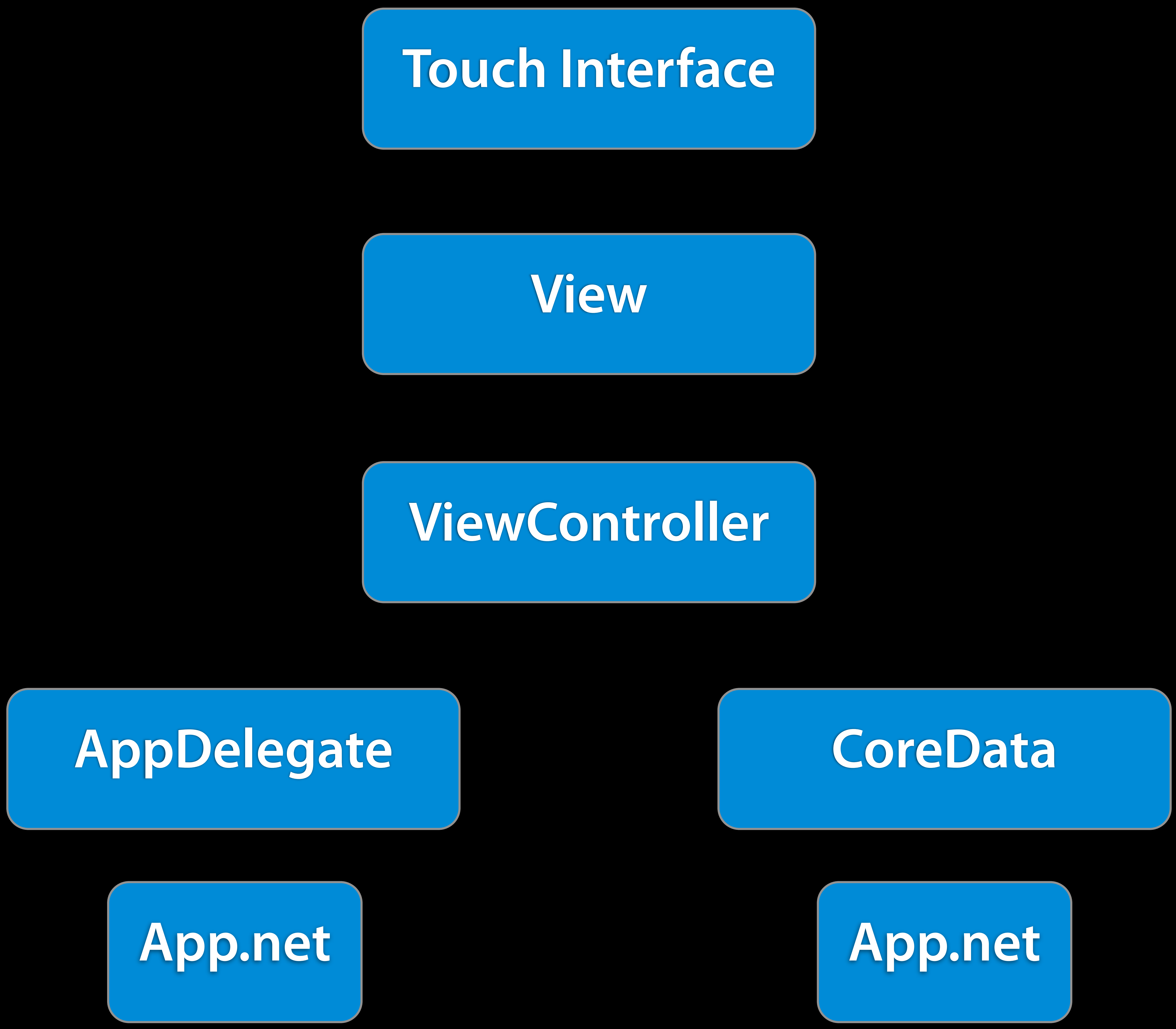
Observer

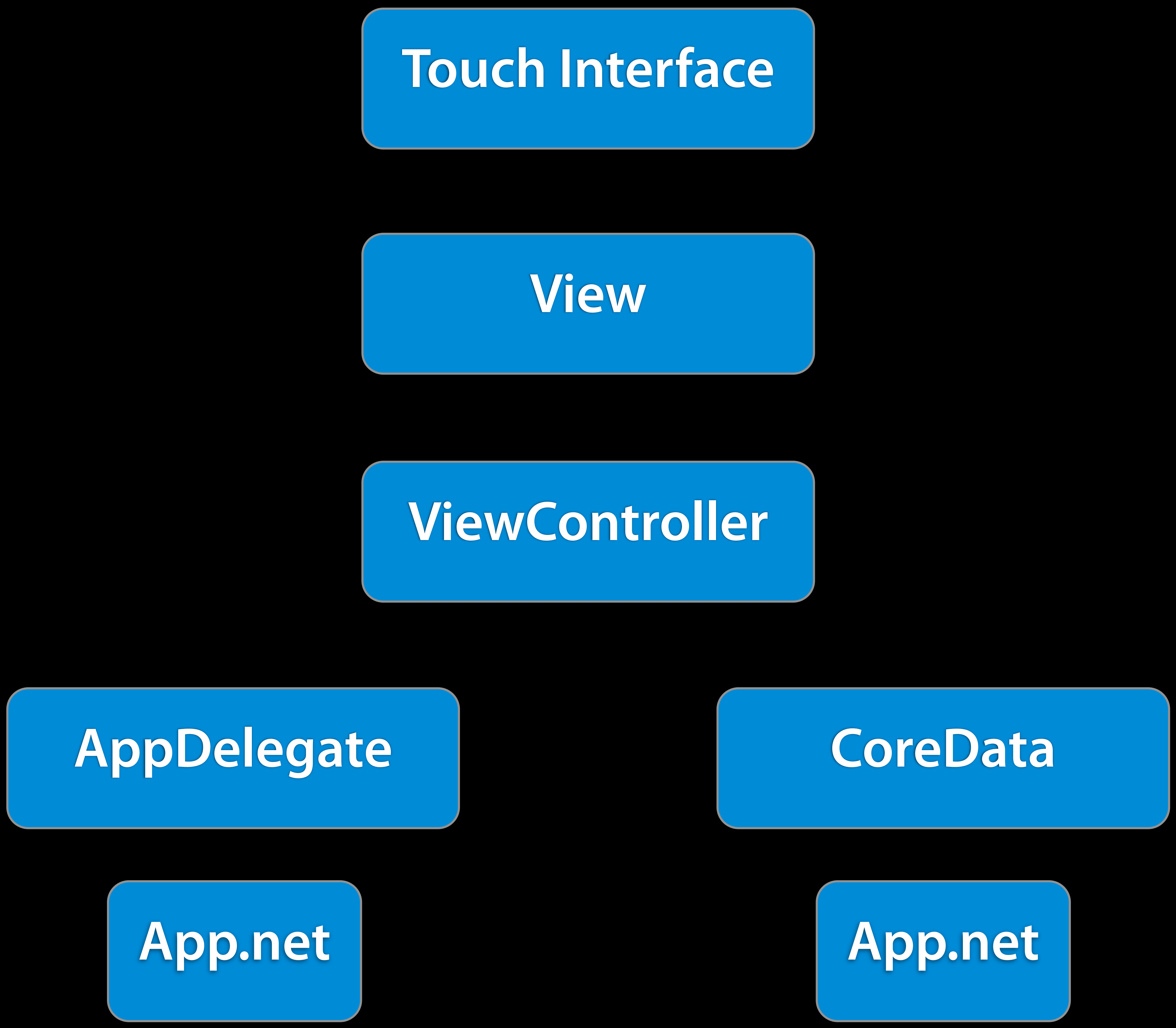
Composite

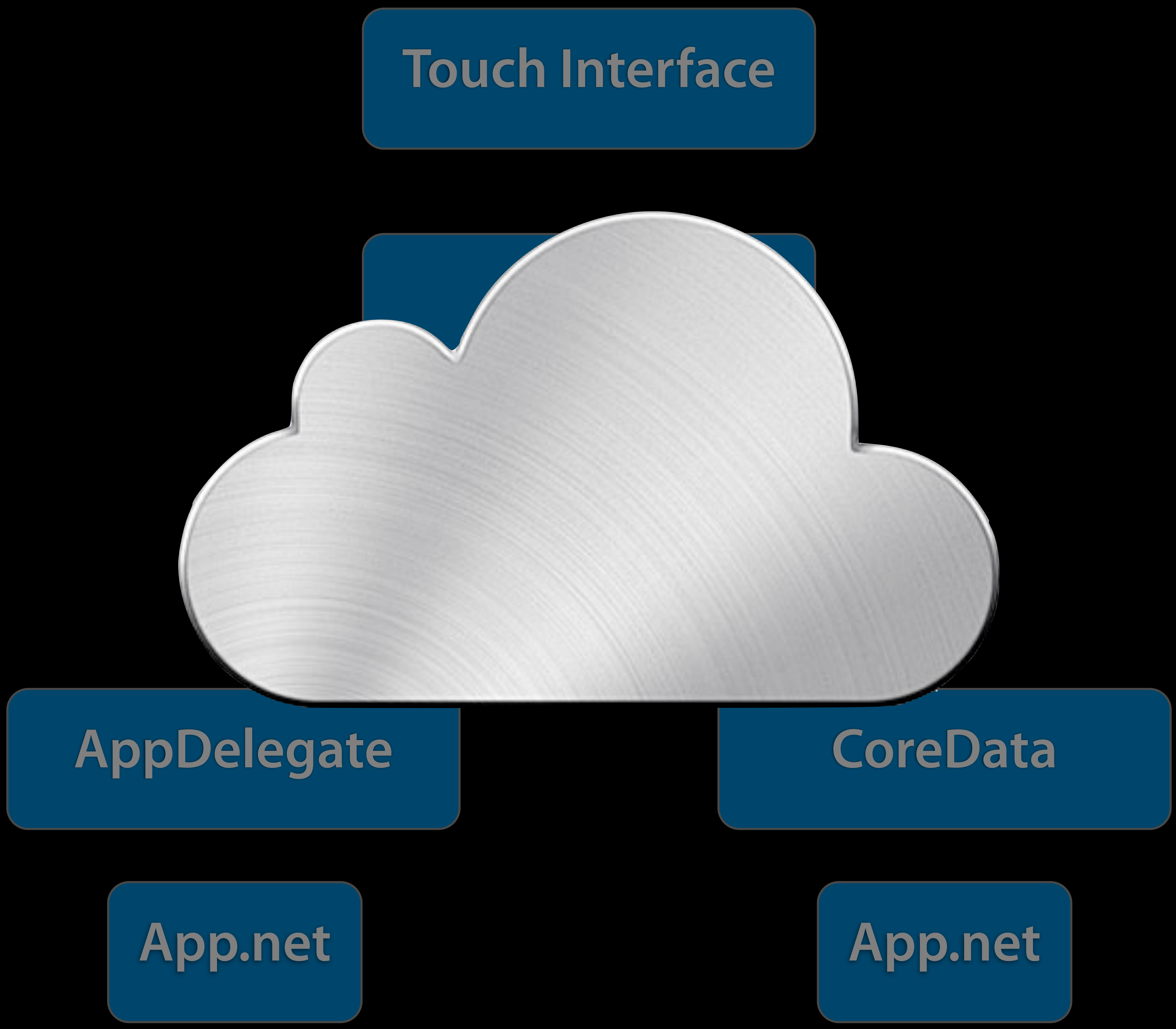
Mediator

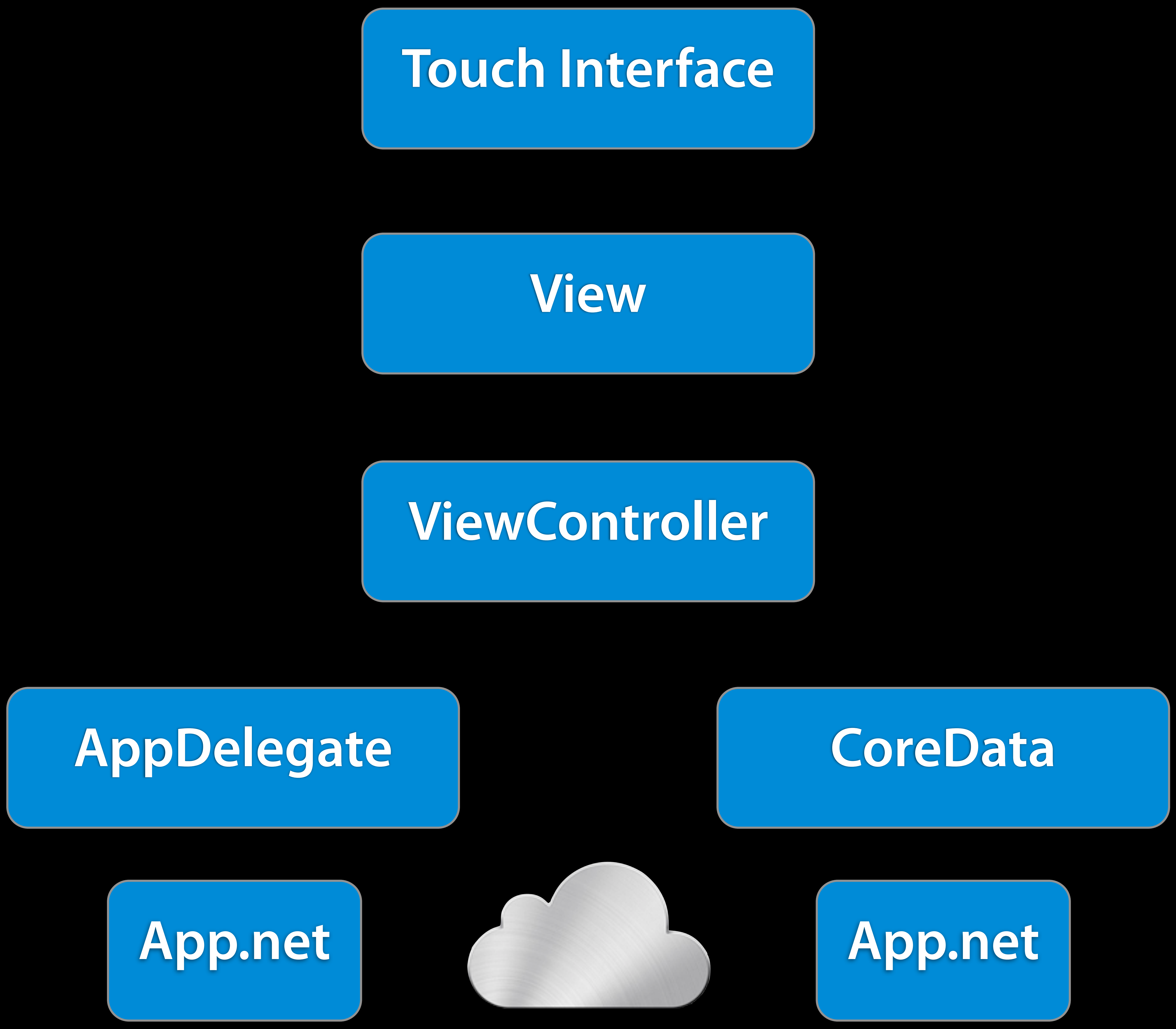
Strategy

MVC is actually composed of quite a few other patterns you might recognize:


Touch Interface
View
ViewController
AppDelegate
CoreData
App.net
App.net


Touch Interface
View
ViewController
AppDelegate
CoreData
App.net
App.net


Touch Interface
AppDelegate
CoreData
App.net
App.net


Touch Interface
View
ViewController
AppDelegate
CoreData
App.net
App.net

Pattern Abuse

There are patterns that are examples of good practices, then there are antipatterns: patterns of solutions that you should not be using

Factory

The factory pattern, over time, leads to a giant logic bottleneck of creation code.

Abstract Factories

Eventually, we end up with factories that make miniature models of factories.
This is because the allocation and initialization of objects are fused together on other languages with the ‘new’ keyword.

Two-Stage Creation

Wednesday, June 19, 13

The original design of Objective C introduced the idea of two stage object creation. First, we alloc the memory, then we call the init method.

```
[[MyClass alloc] init]
```

This lets us eliminate the factory pattern in most cocoa code by calling the correct setup code for a particular instance. Then to create the correct type, we use...

NSClassFromString

Wednesday, June 19, 13

In Cocoa, we can use NSClassFromString to achieve a result largely eliminating the factory pattern.

[NSClassFromString(@”MyClass”) alloc] init]

The Designated initializer helps ensure the correct initialization occurs
http://developer.apple.com/library/ios/#documentation/general/conceptual/DevPedia-CocoaCore/MultipleInitializers.html

More explanation in slides

~~Singletons~~

Wednesday, June 19, 13

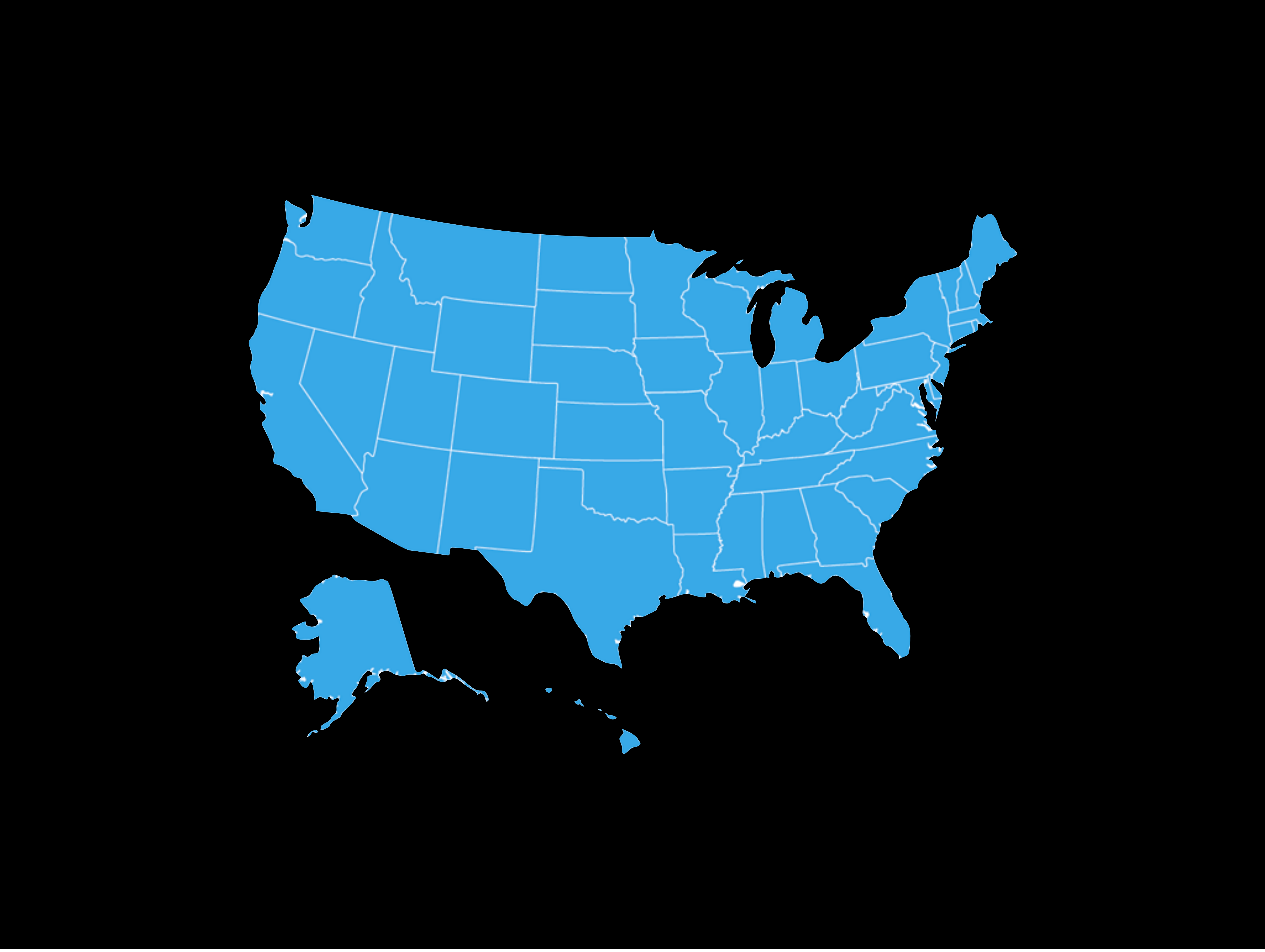
Singletons are the most well known, and worst used pattern. Don't use them unless you have a REALLY good reason.

- Singletons are a single instance of an object, which has consequences:
-- A global state
-- Not always thread-safe
-- No way to control object lifecycle
-- Tight coupling of your code
-- Doesn't always accurately reflect system model (there can be more than one instance)

However, if you ARE going to use them (despite my sound advice), here is the best way to do so, without shooting yourself in the foot:
-- Use them in your classes as if they were instances of objects
-- Inject a singleton as a parameter
-- remember to make sure the variable name has a meaning beyond the fact that it's a singleton

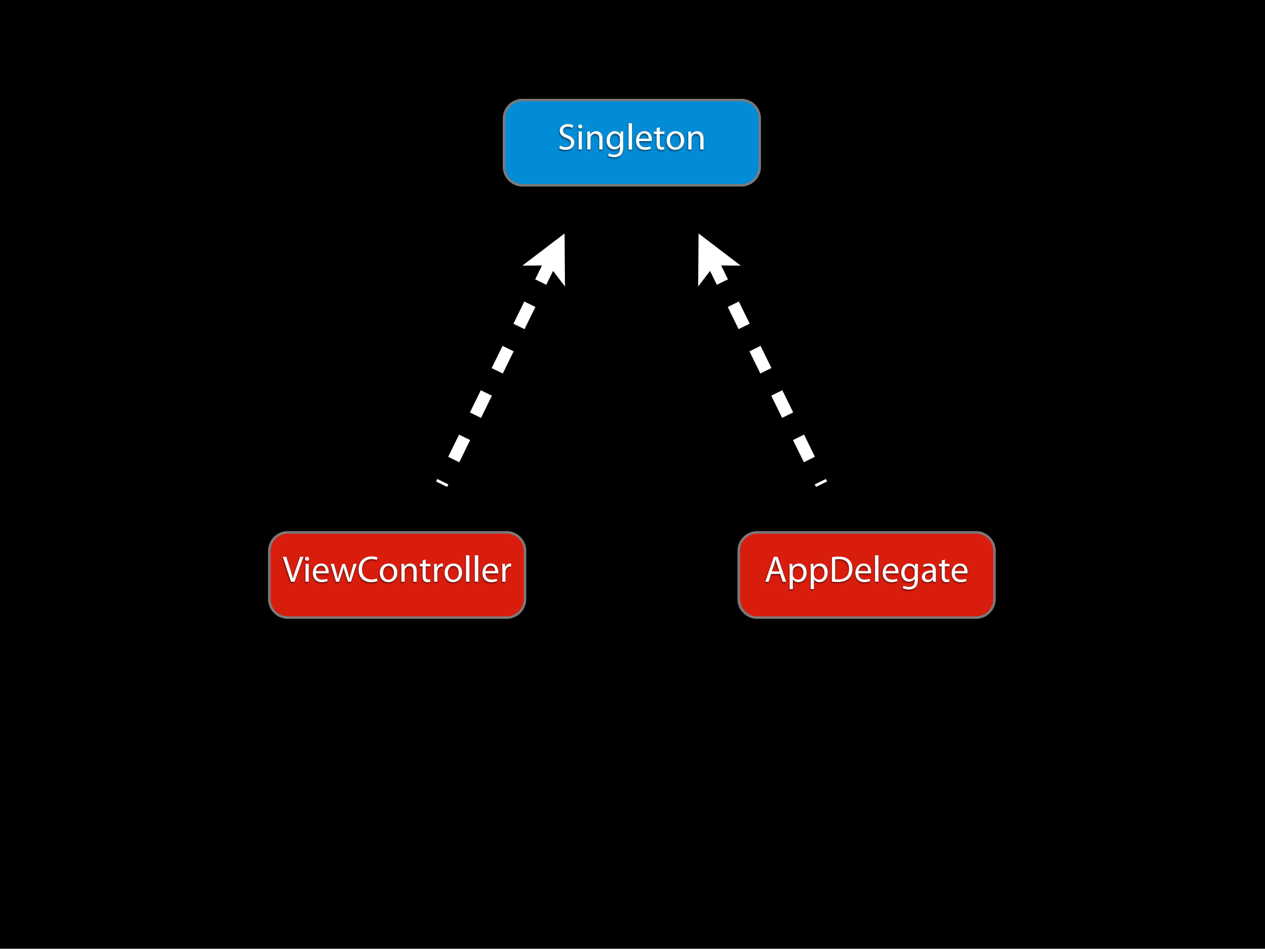
Wednesday, June 19, 13

Singletons are just fancy global variables.



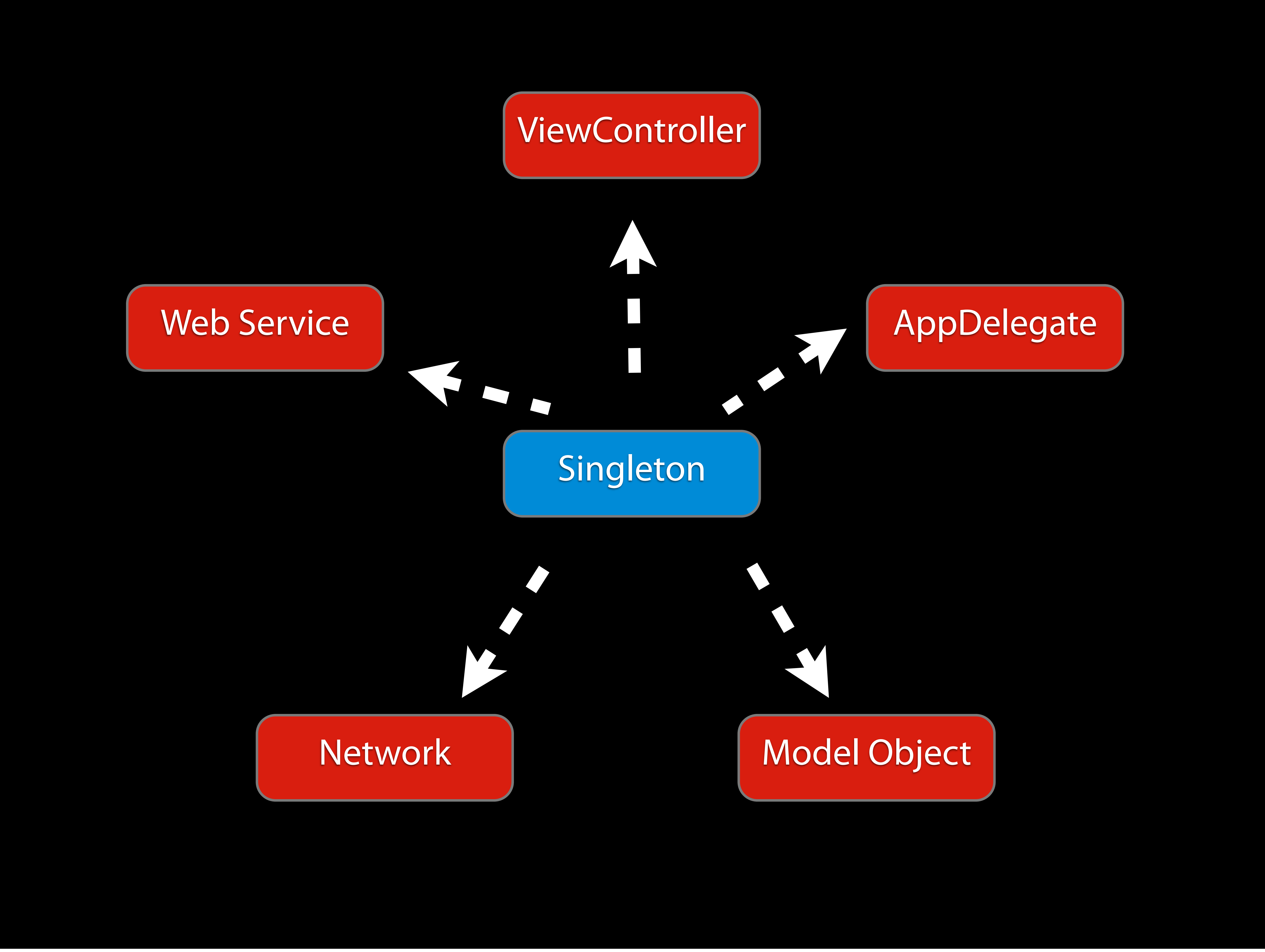
Wednesday, June 19, 13

Global variables mean an application wide shared state, which is easy to corrupt across threads.

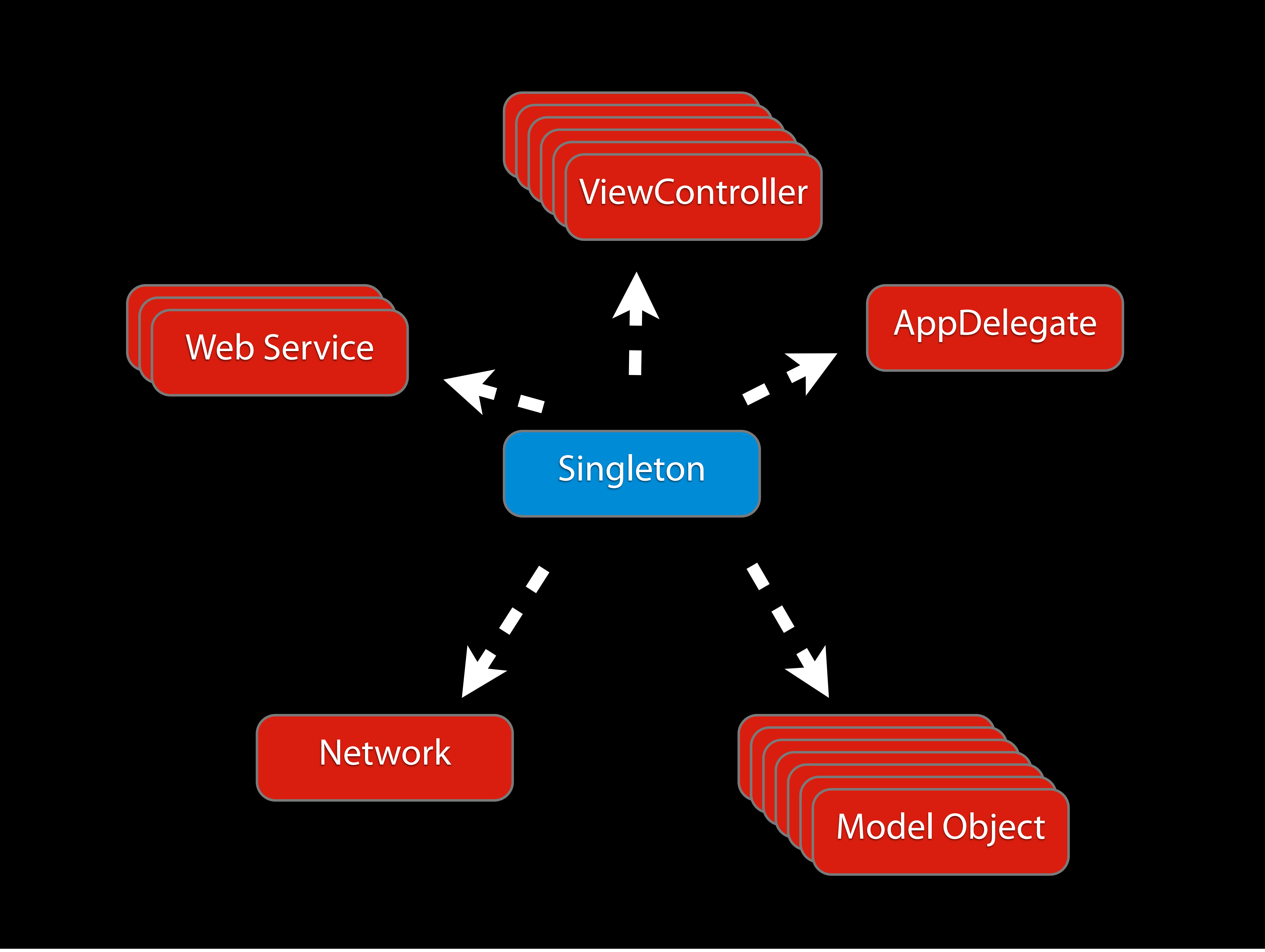


Wednesday, June 19, 13

Your app starts off easy enough using a singleton in the right places



But then you need it in other places



Wednesday, June 19, 13

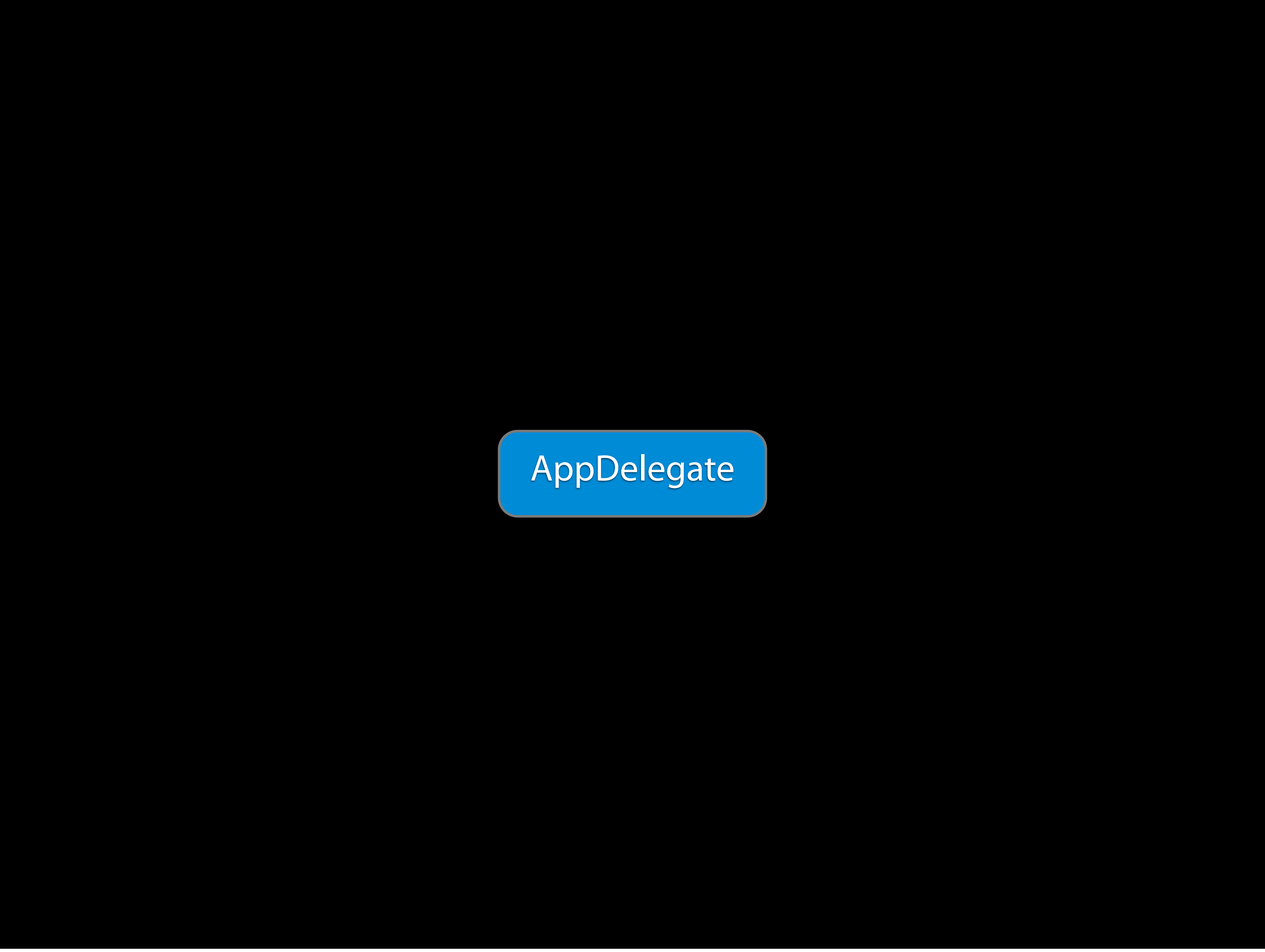
Then you end up with Tight coupling, singletons referenced and used everywhere.

Singleton

Wednesday, June 19, 13

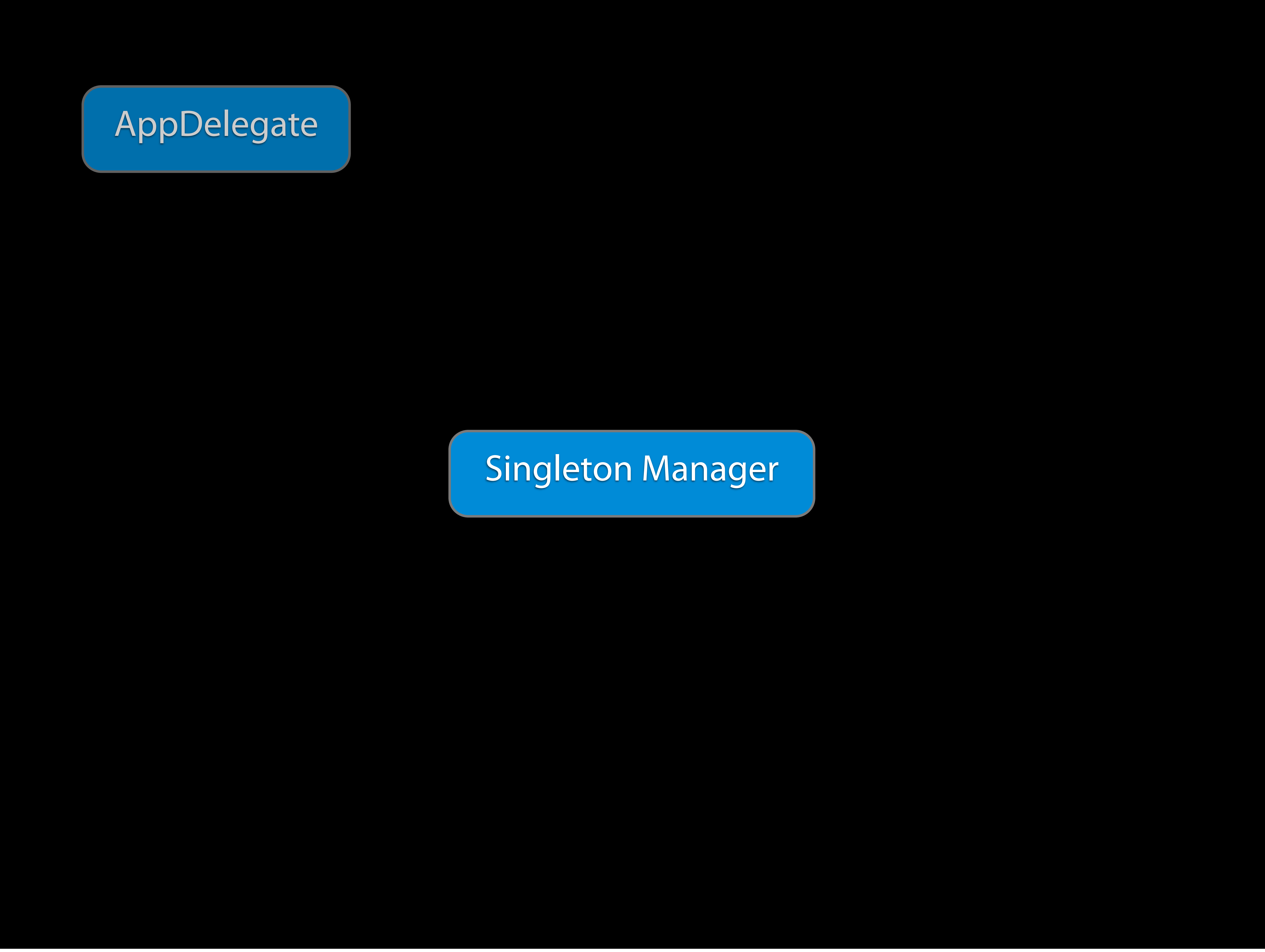
Singletons can be around for the lifetime of your app. This is not always ideal as it can eat away resources.

Singleton

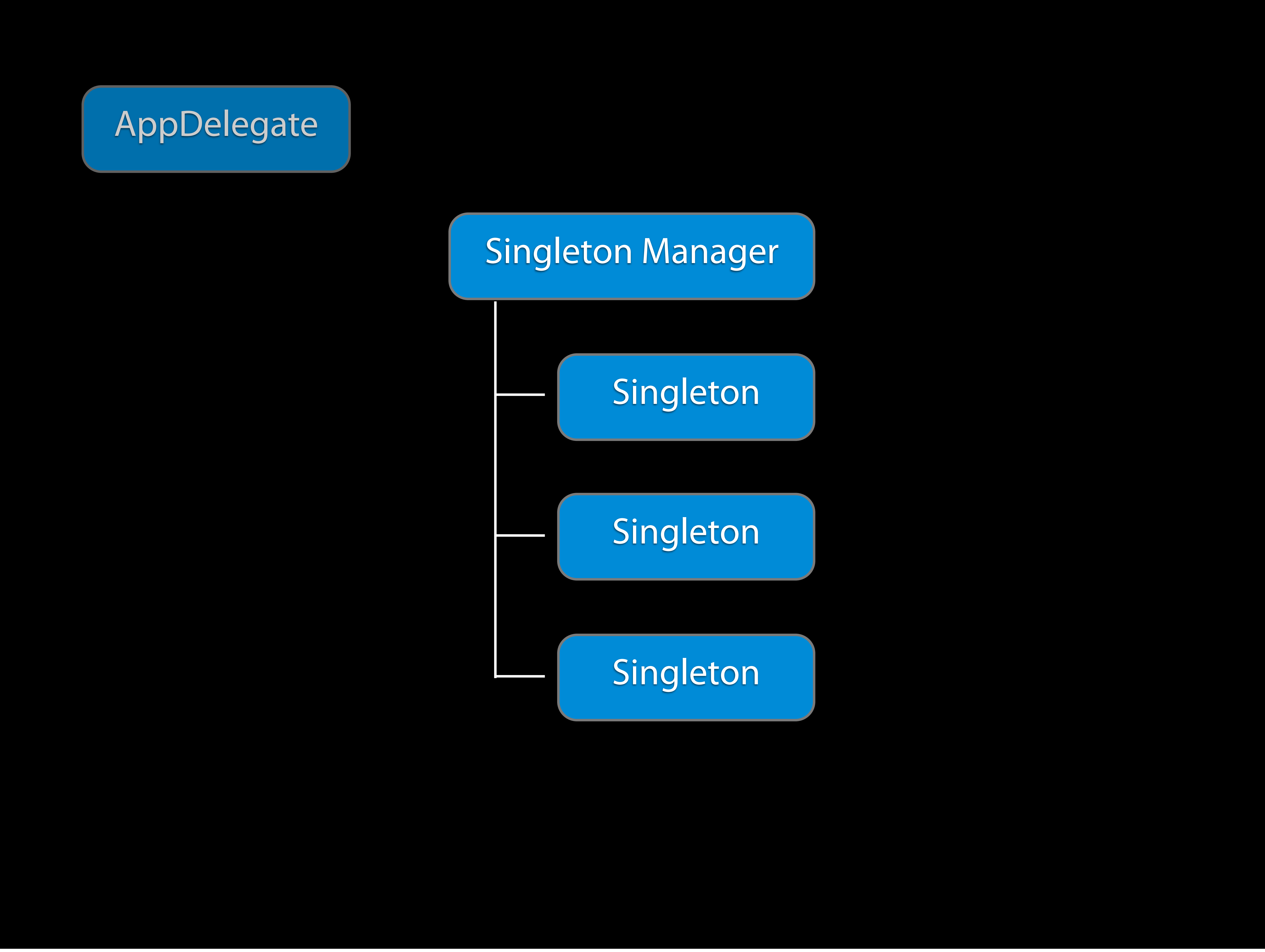


Wednesday, June 19, 13

Singleton Lifecycle Manager...is also a singleton...sort of

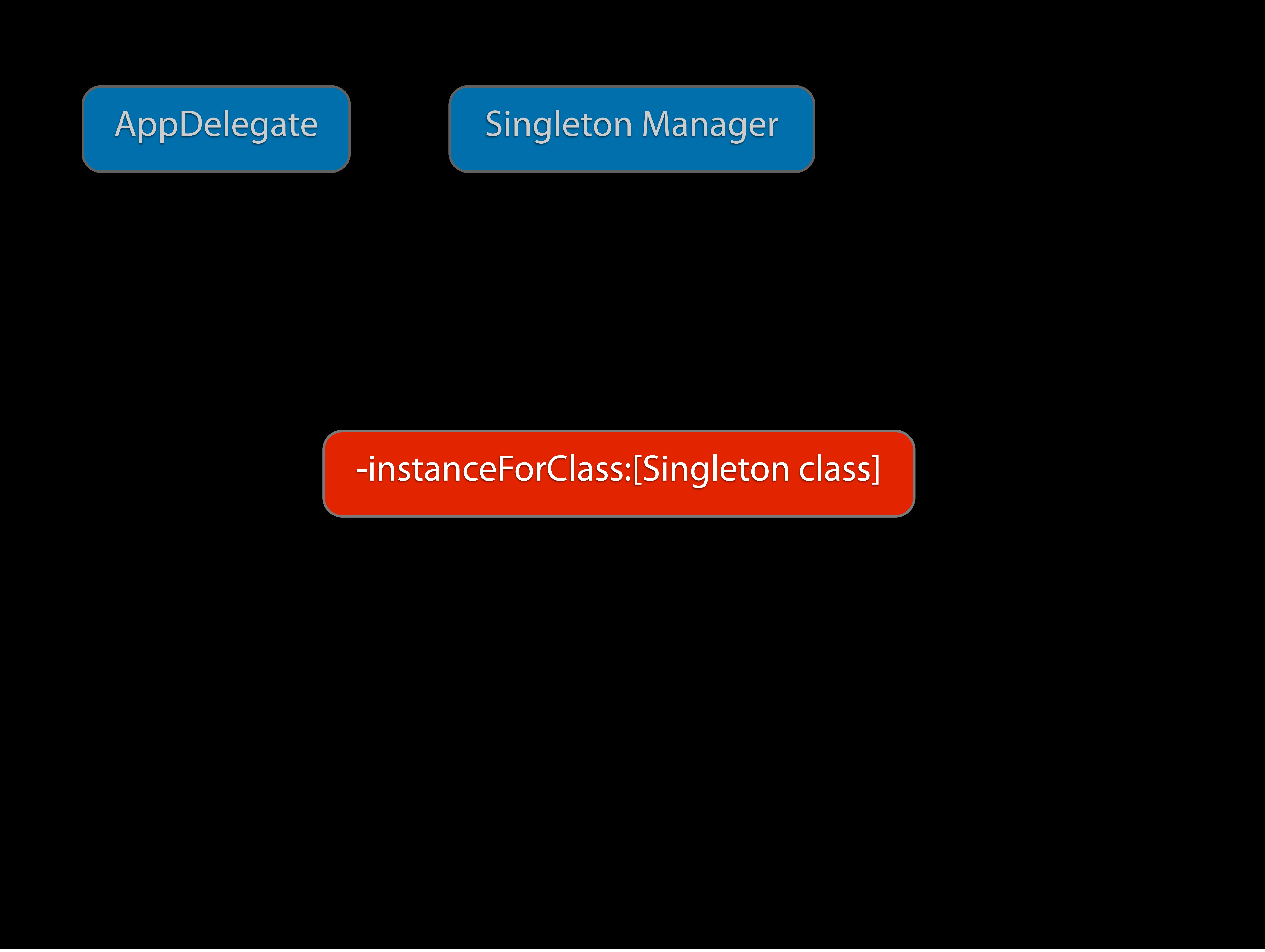


Singleton Lifecycle Manager...is also a singleton...sort of



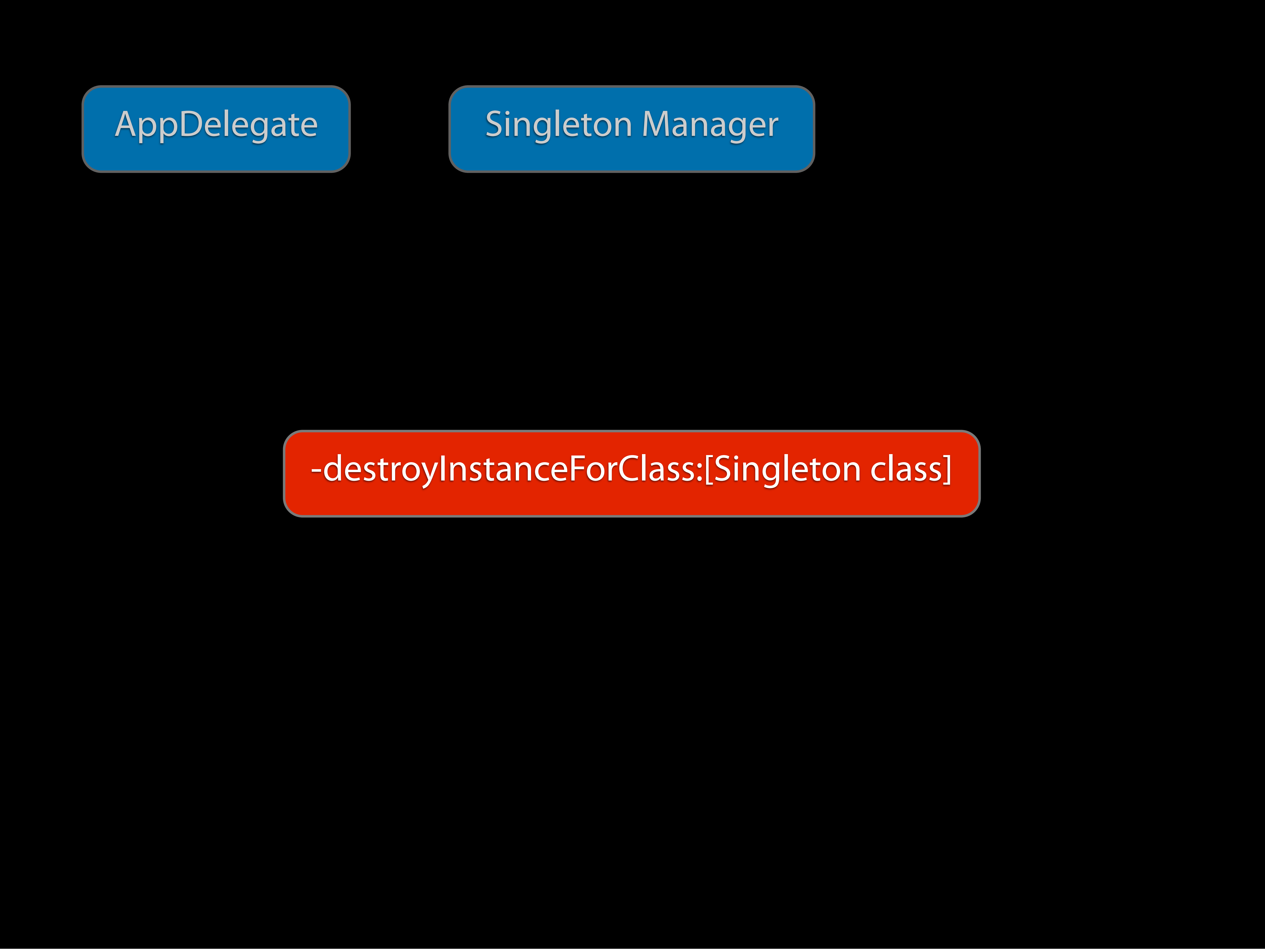
Wednesday, June 19, 13

Singleton Lifecycle Manager...is also a singleton...sort of



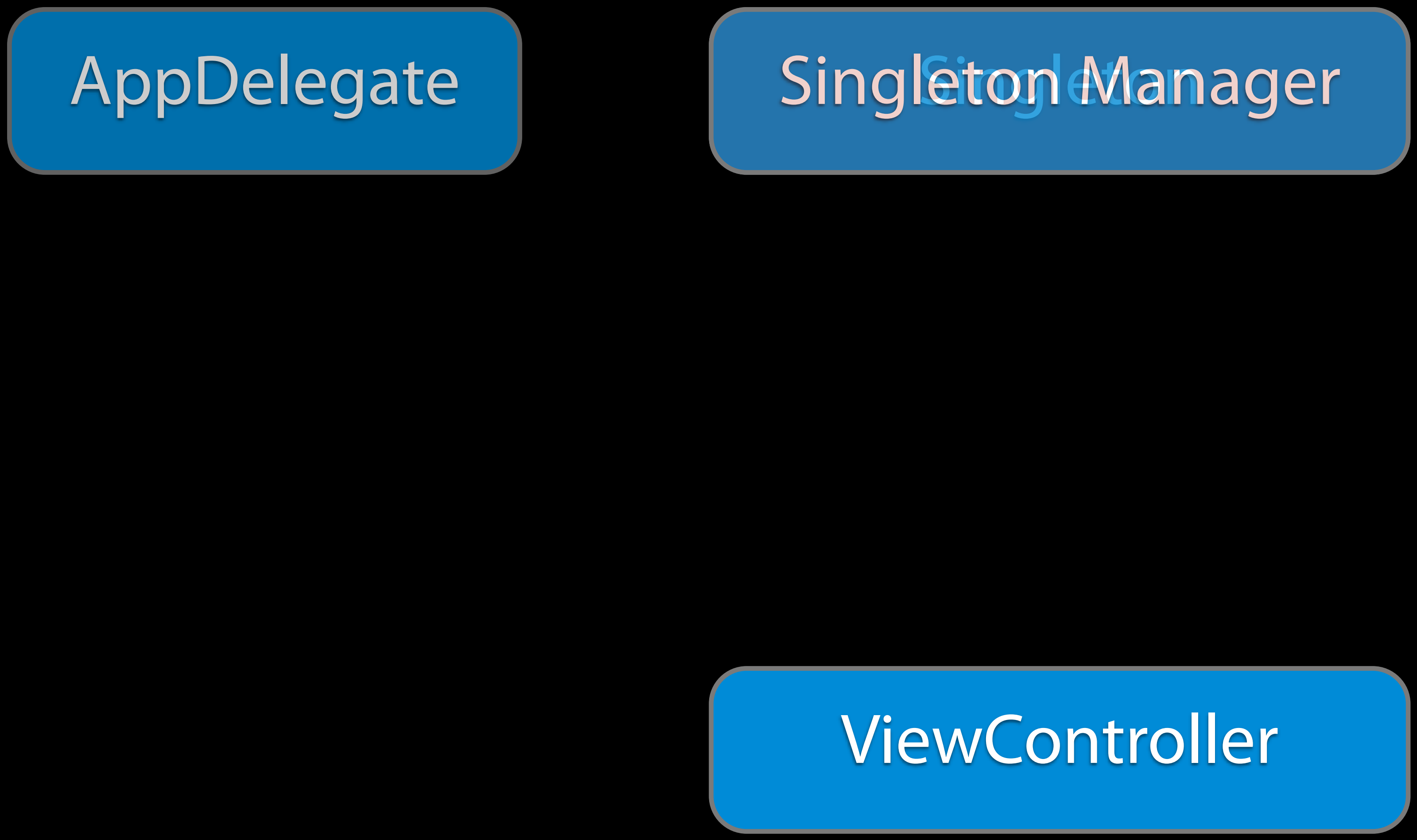
Wednesday, June 19, 13

Singleton Lifecycle Manager...is also a singleton...sort of



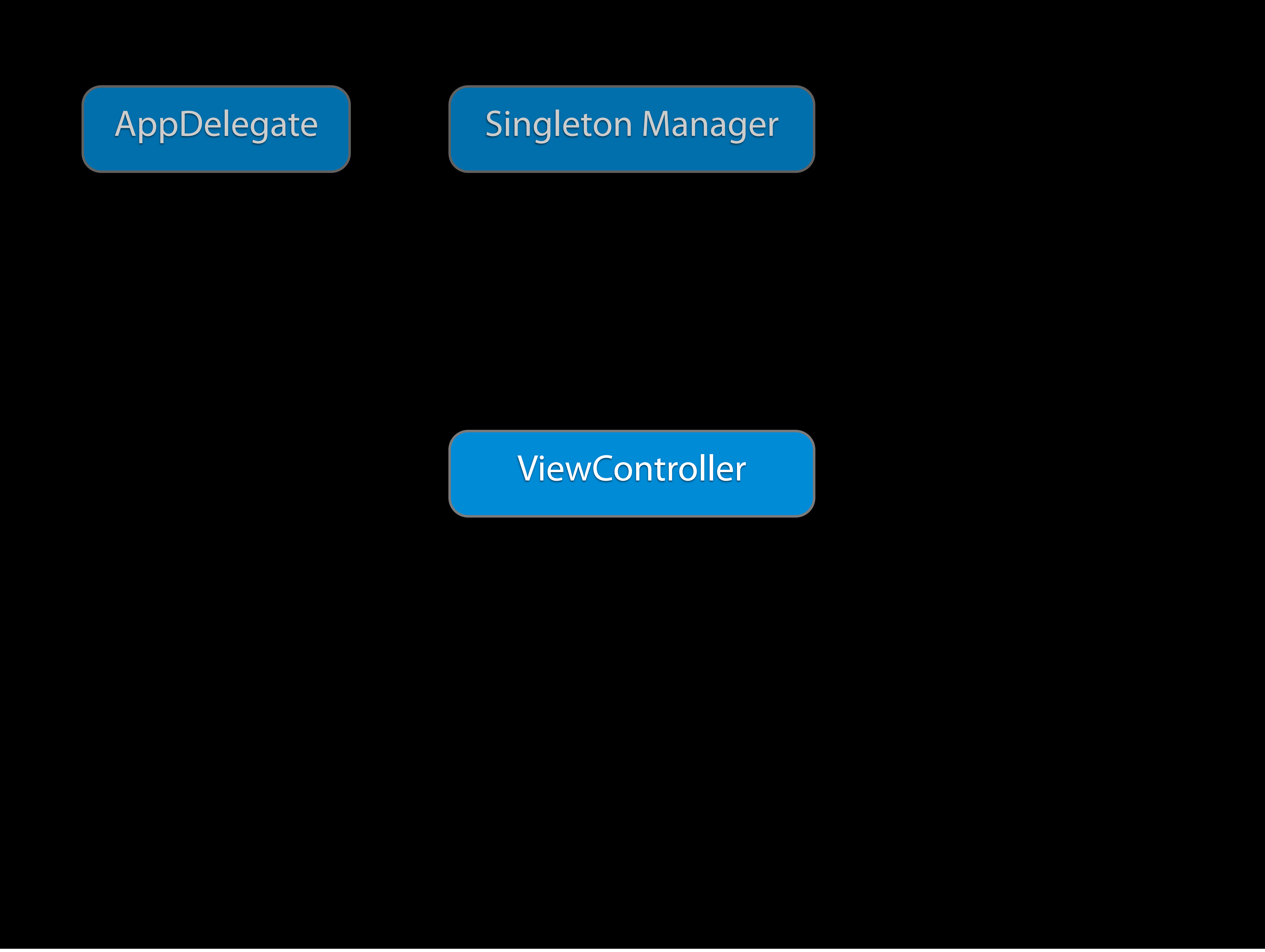
Wednesday, June 19, 13

Singleton Lifecycle Manager...is also a singleton...sort of



Wednesday, June 19, 13

And when you want to use a singleton, you want to inject it into your class



Wednesday, June 19, 13

And when you want to use a singleton, you want to inject it into your class

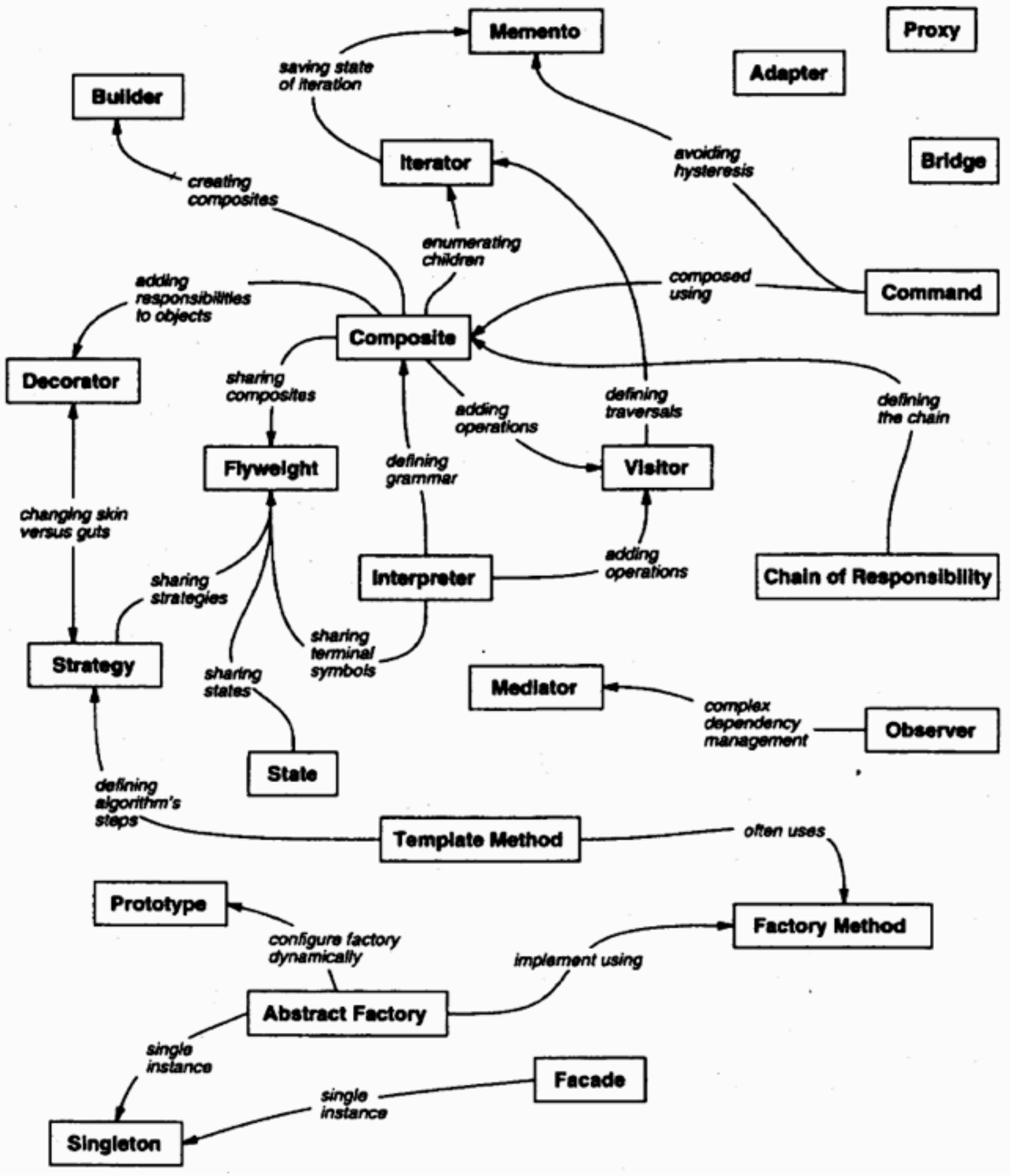
ServiceLocator

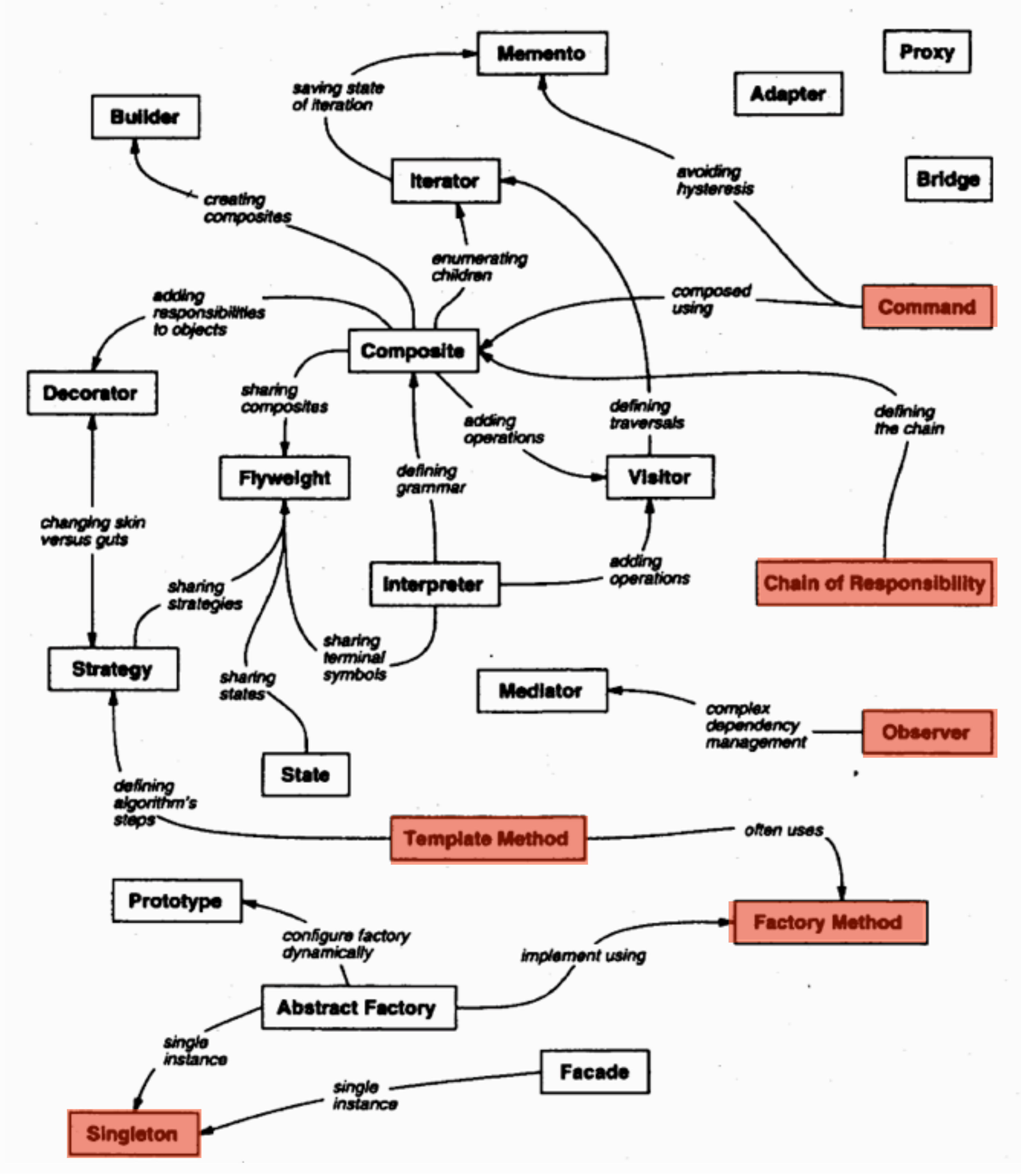
Classes are objects too...

Which means they are singletons

```
@property Class dateCreator;
```

```
[self.dateCreator date];
```


Memento
Proxy
Adapter
Builder
saving state of iteration
Iterator
avoiding hysteresis
Bridge
creating composites
enumerating children
composed using
Command
adding responsibilities to objects
Composite
Decorator
sharing composites
adding operations
defining traversals
defining the chain
Flyweight
defining grammar
Visitor
changing skin versus guts
sharing strategies
Interpreter
adding operations
Chain of Responsibility
sharing terminal symbols
Strategy
sharing states
Mediator
complex dependency management
Observer
State
defining algorithm's steps
Template Method
often uses
Prototype
configure factory dynamically
implement using
Factory Method
Abstract Factory
single instance
Facade
single instance
Singleton


Memento
Proxy
Adapter
Builder
saving state of iteration
Iterator
avoiding hysteresis
Bridge
creating composites
enumerating children
adding responsibilities to objects
composed using
Command
Composite
Decorator
sharing composites
adding operations
defining traversals
defining the chain
Flyweight
defining grammar
Visitor
changing skin versus guts
Interpreter
adding operations
Chain of Responsibility
sharing strategies
Strategy
sharing terminal symbols
sharing states
Mediator
complex dependency management
Observer
State
defining algorithm's steps
Template Method
often uses
Prototype
Factory Method
configure factory dynamically
implement using
Abstract Factory
single instance
Facade
single instance
Singleton

The dirty little secret about this talk: These patterns aren’t mobile specific, they apply to any well built application. But, these patterns are the one’s I’ve found most common in the apps I’ve see, and the code I’ve written, as such, they qualify as patterns for mobile apps. (Trying to reframe design patterns into a modern world of mobile computing)

Think about when the Design Patterns book was written. A time when 486 was the top of the line processor and memory maxed out on your machine at 4 MB. Megabytes! Your iPhone has a 1GHz processor and at least 256 MB of RAM. The constraints were there, and these patterns worked. These will work well on mobile applications.

Another note about patterns is that these are not always going to be implemented as a generic, reusable library or framework. Some languages, like C++, Java and C+ support concepts like Templates and Generics that may facilitate the implementation or use of these patterns.

Head First Design Patterns
Patterns of Enterprise Application Architecture
Service Design Patterns
Enterprise Integration Patterns
Patterns for Parallel Programming
Java Design Patterns A Tutorial
Design Patterns in C#
Analysis Patterns
Cocoa Design Patterns
Design Patterns Explained

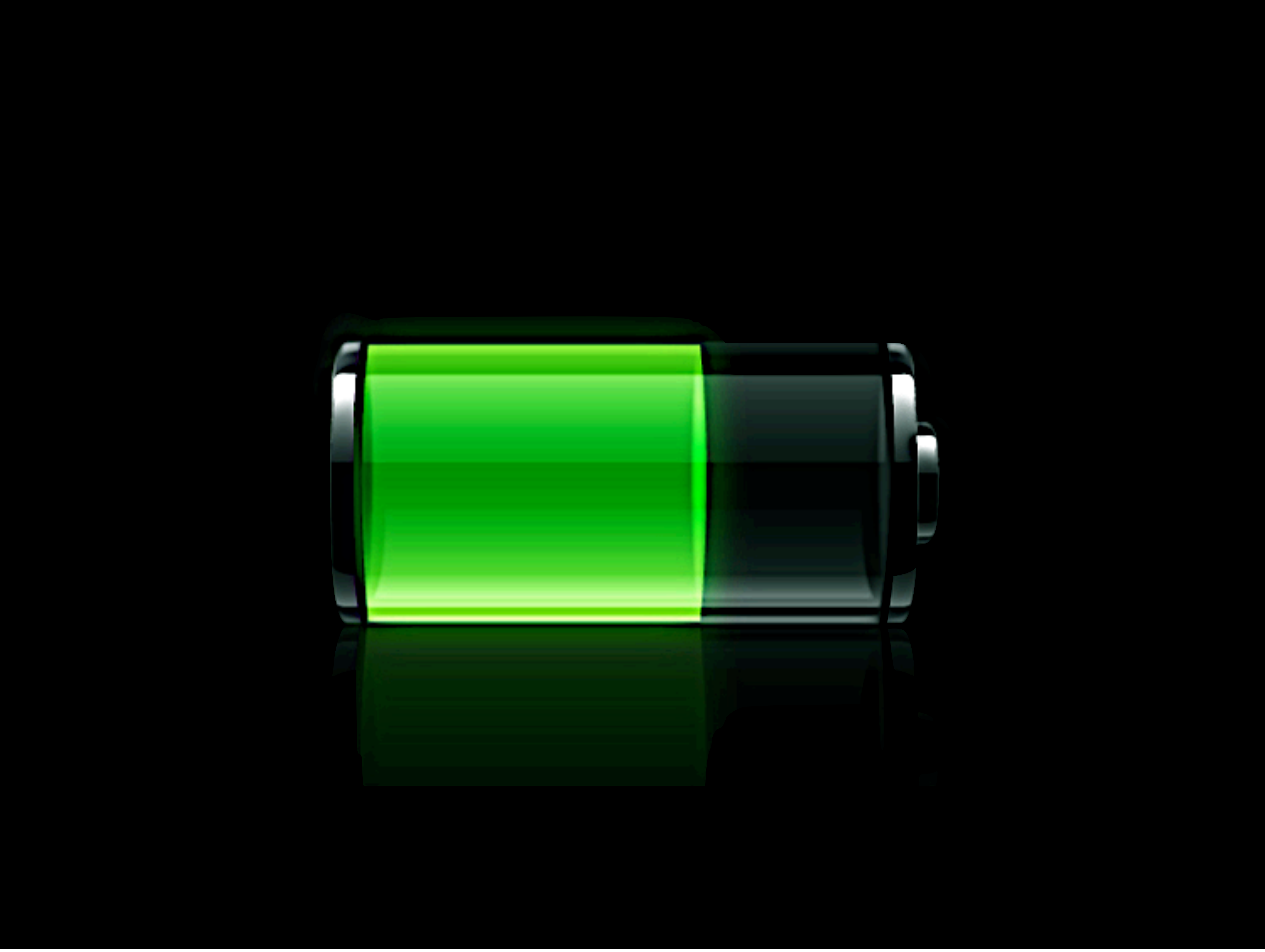
Wednesday, June 19, 13

Since the original GoF book was written, many new books delving into other areas of patterns have emerged.

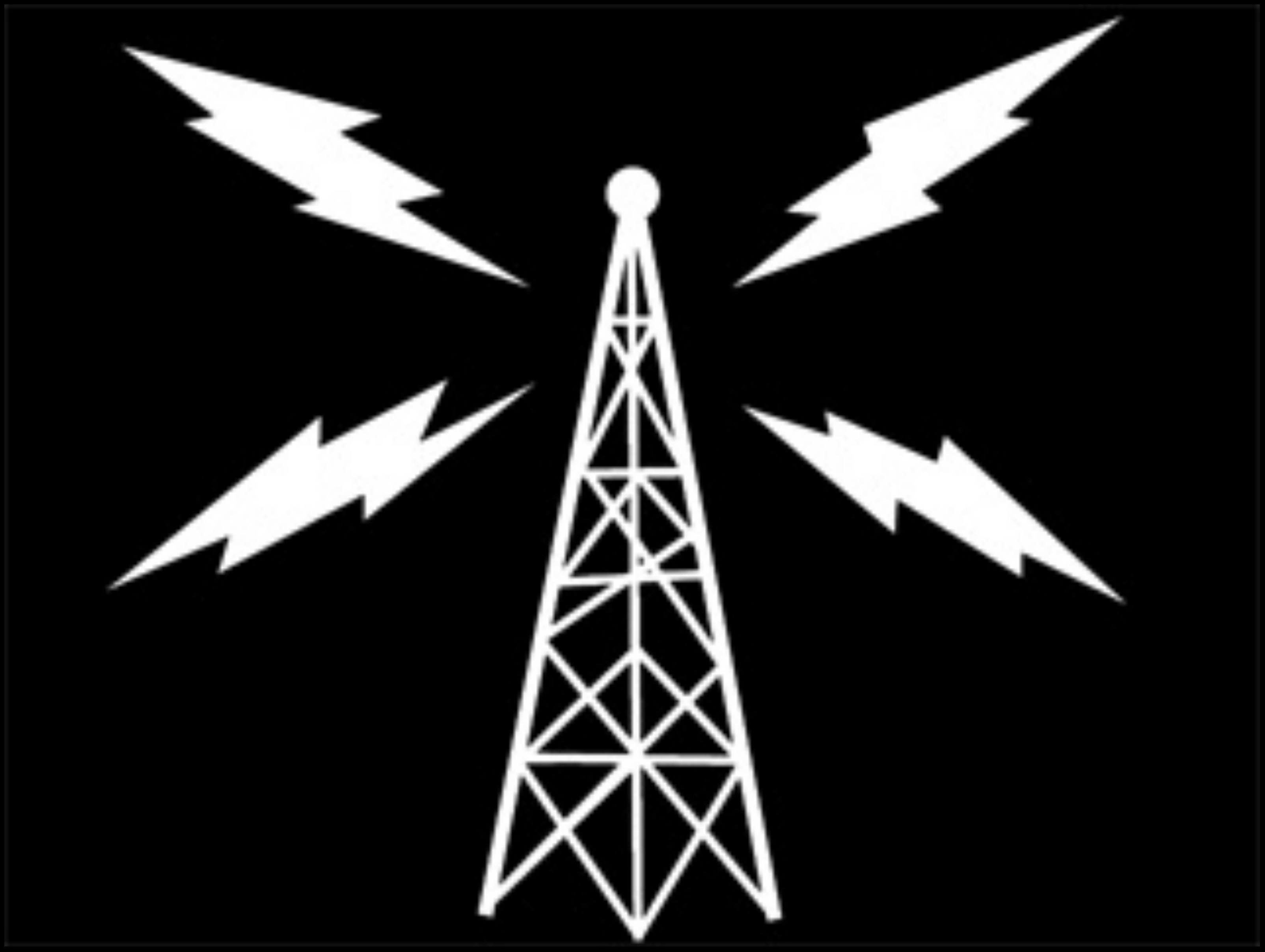
What's old is new again

These patterns existed long ago, in a time of fairly limited system resources. Many of these limitations parallel those of mobile devices over the past 5 years.

(The terracota warriors, one of the great discoveries of the past)
There are still many patterns to be discovered that are mobile specific
Patterns related to:
- Managing battery power
- Network radio management
- concurrency

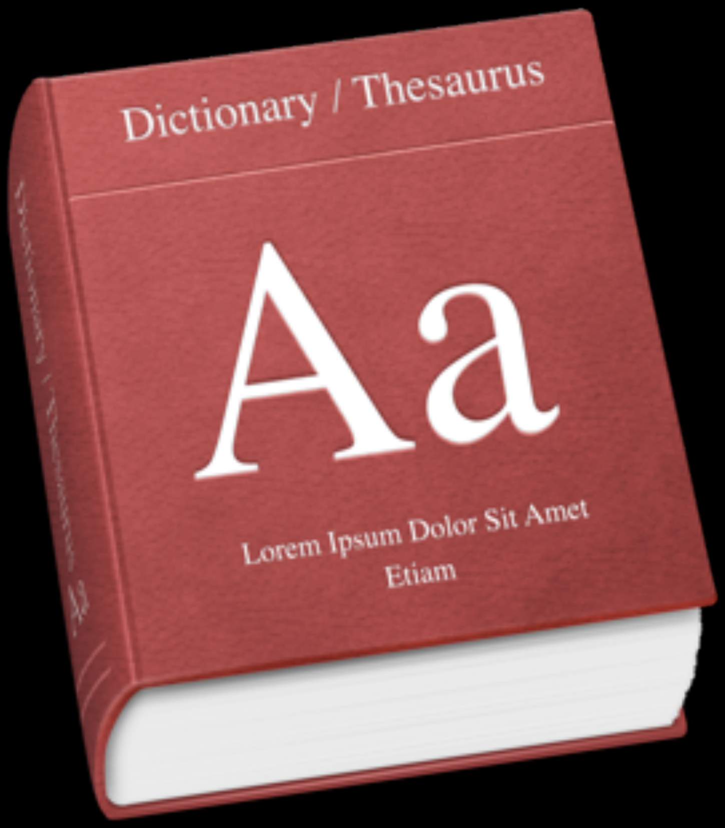


Battery Power
- More laptops can be considered mobile devices these days, and algorithms that consume less power will be more important over time.



Radio power/transmission consolidation

Concurrency Patterns



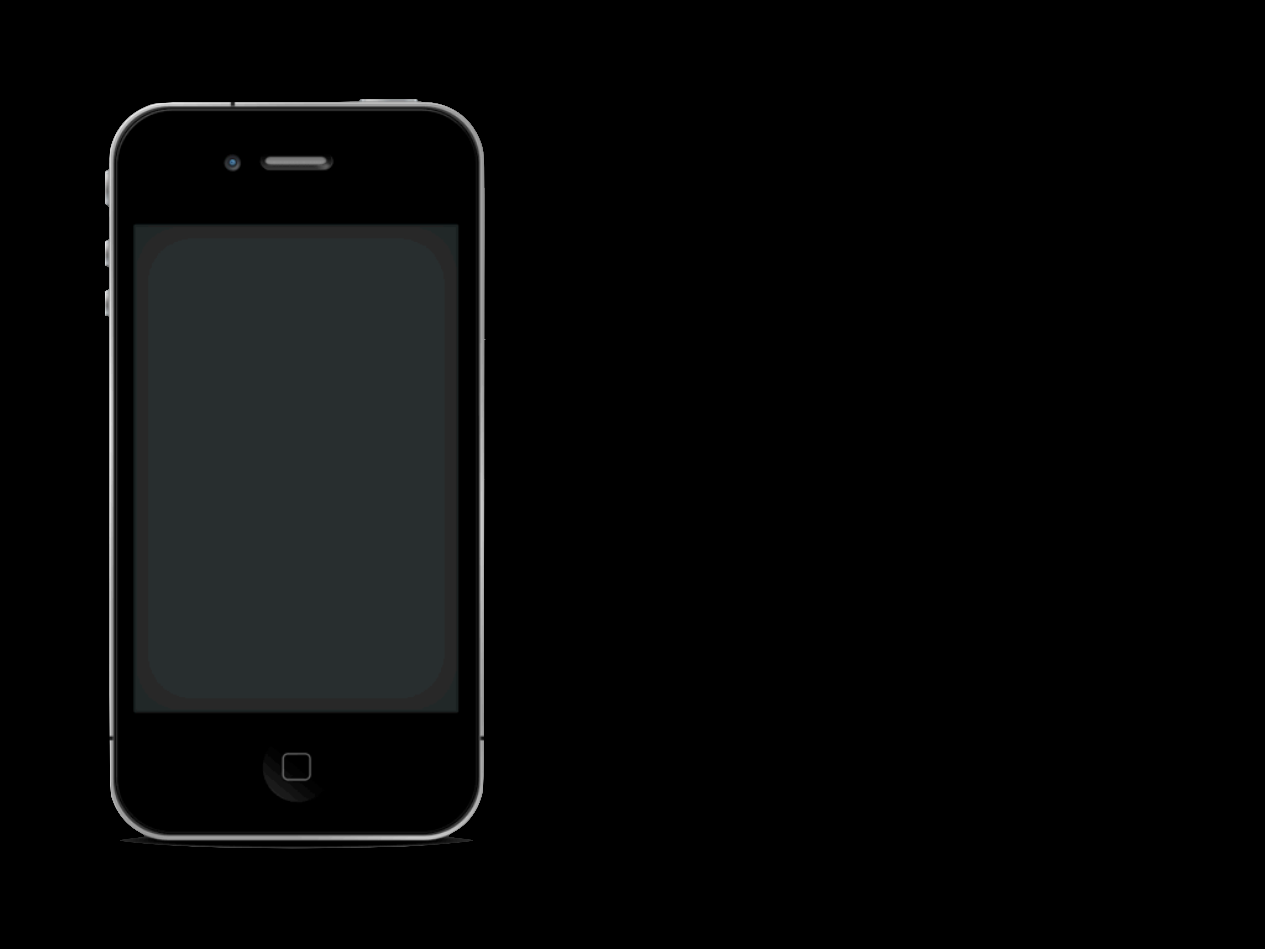
Auto Purging Cache
- Dictionary, key/value store
- Keeps track of cached items/cost of each item based on heuristic
- When memory is low, cache is auto-purged

NSCache

Wednesday, June 19, 13

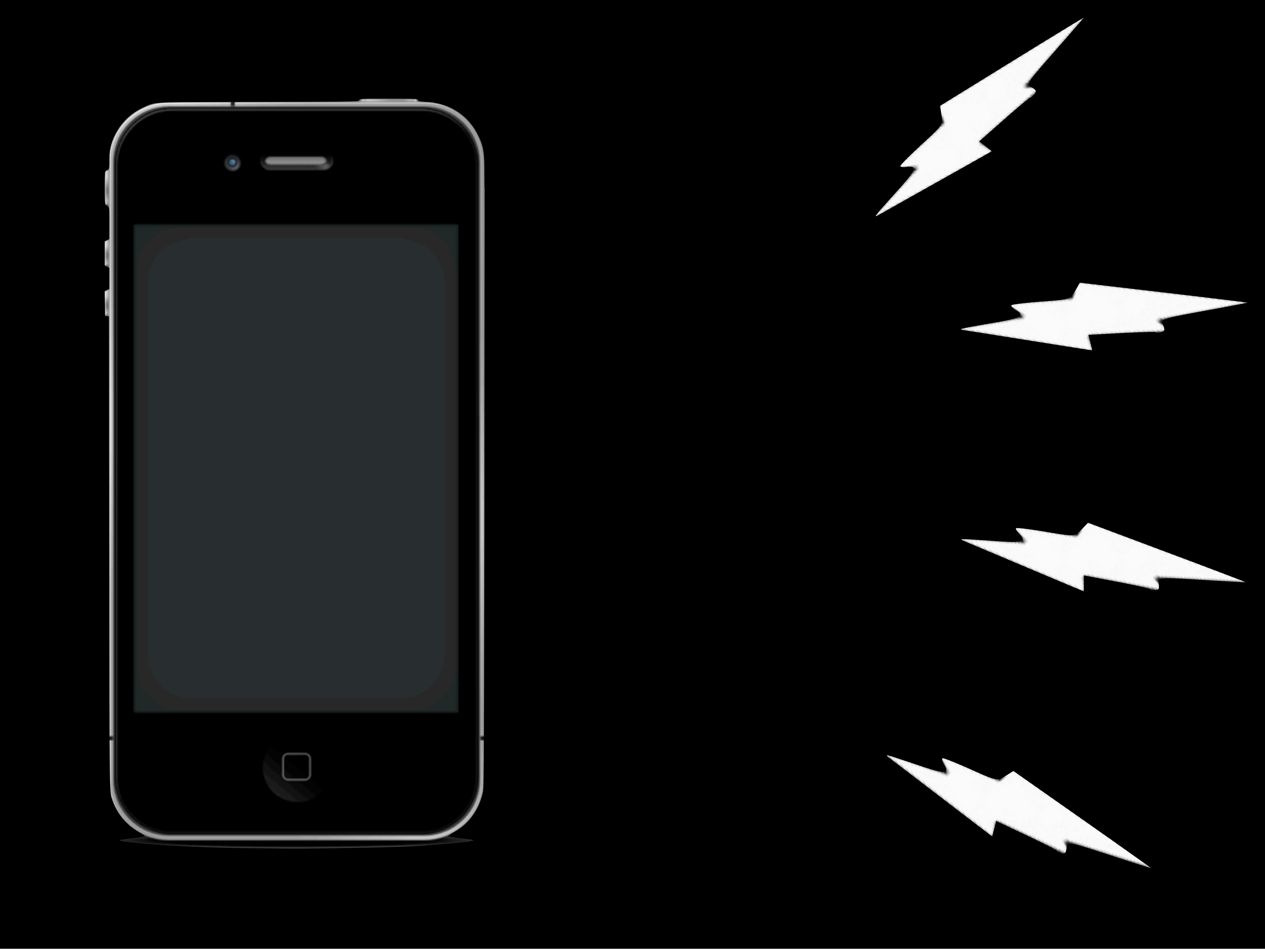
On iOS and Mac, this pattern, like many discussed today, is already part of the Cocoa framework.

-[cache setTotalCostLimit:]
-[cache setValue:forKey:cost:]

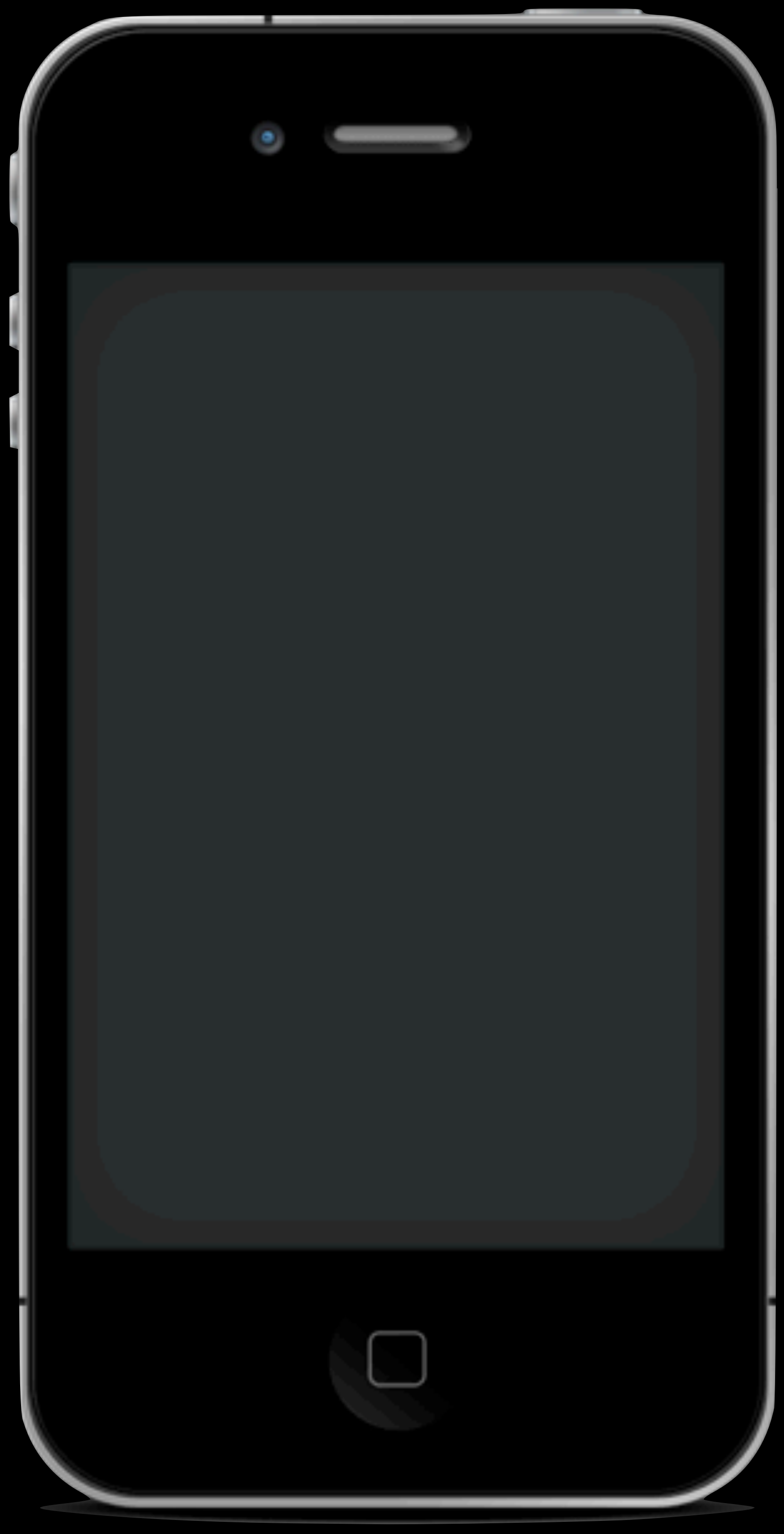


Wednesday, June 19, 13

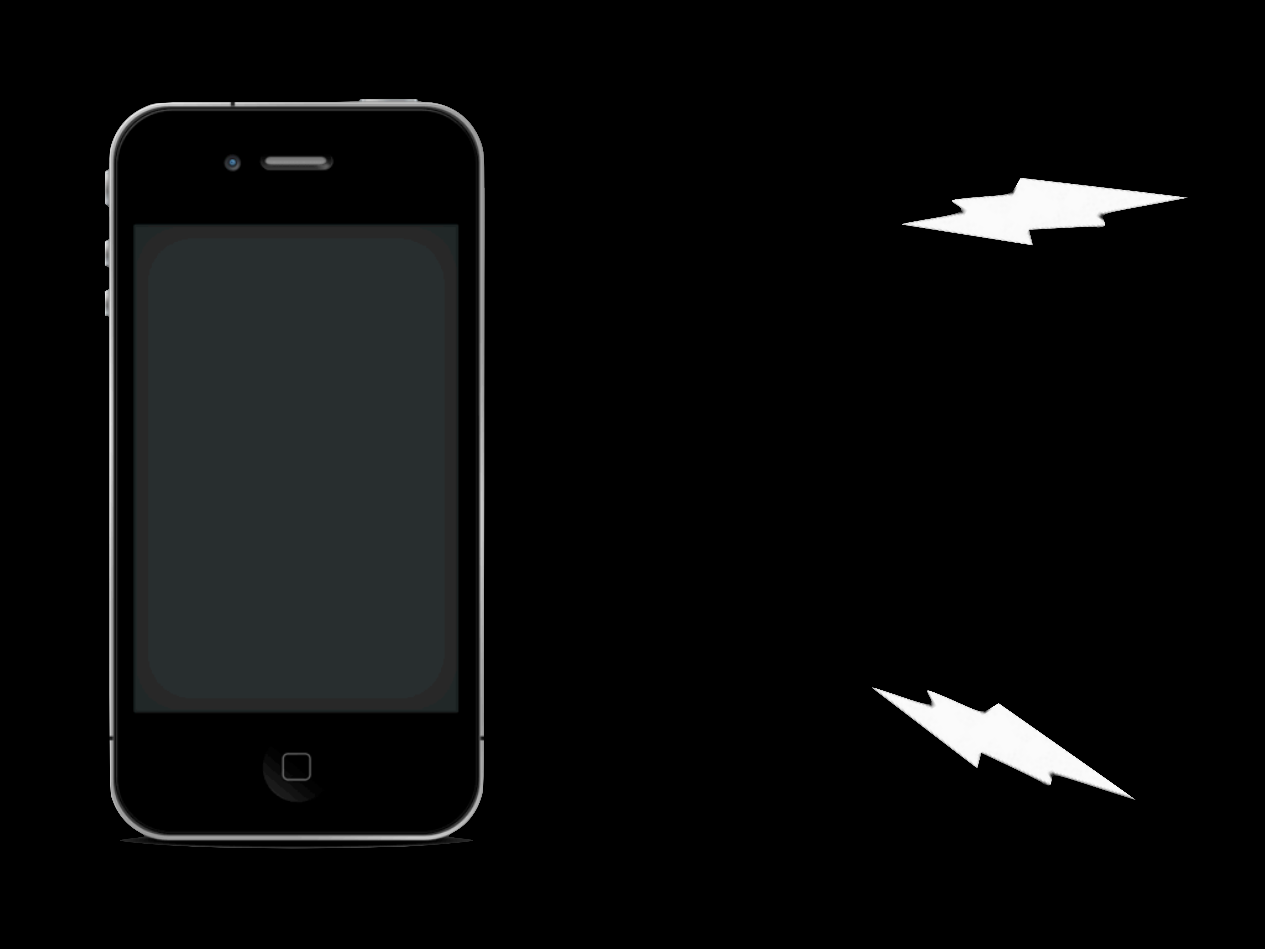
Reachability
- One such pattern is to reconfigure your network/communication stack to behave differently based on different network service levels. Reconfigure your app to respond to the changes in network, on disconnection or in low bandwidth scenarios. Use a strategy pattern for the different network configurations, and the observer pattern to watch for the changes.



Reachability
- One such pattern is to reconfigure your network/communication stack to behave differently based on different network service levels. Reconfigure your app to respond to the changes in network, on disconnection or in low bandwidth scenarios. Use a strategy pattern for the different network configurations, and the observer pattern to watch for the changes.



Reachability
- One such pattern is to reconfigure your network/communication stack to behave differently based on different network service levels. Reconfigure your app to respond to the changes in network, on disconnection or in low bandwidth scenarios. Use a strategy pattern for the different network configurations, and the observer pattern to watch for the changes.



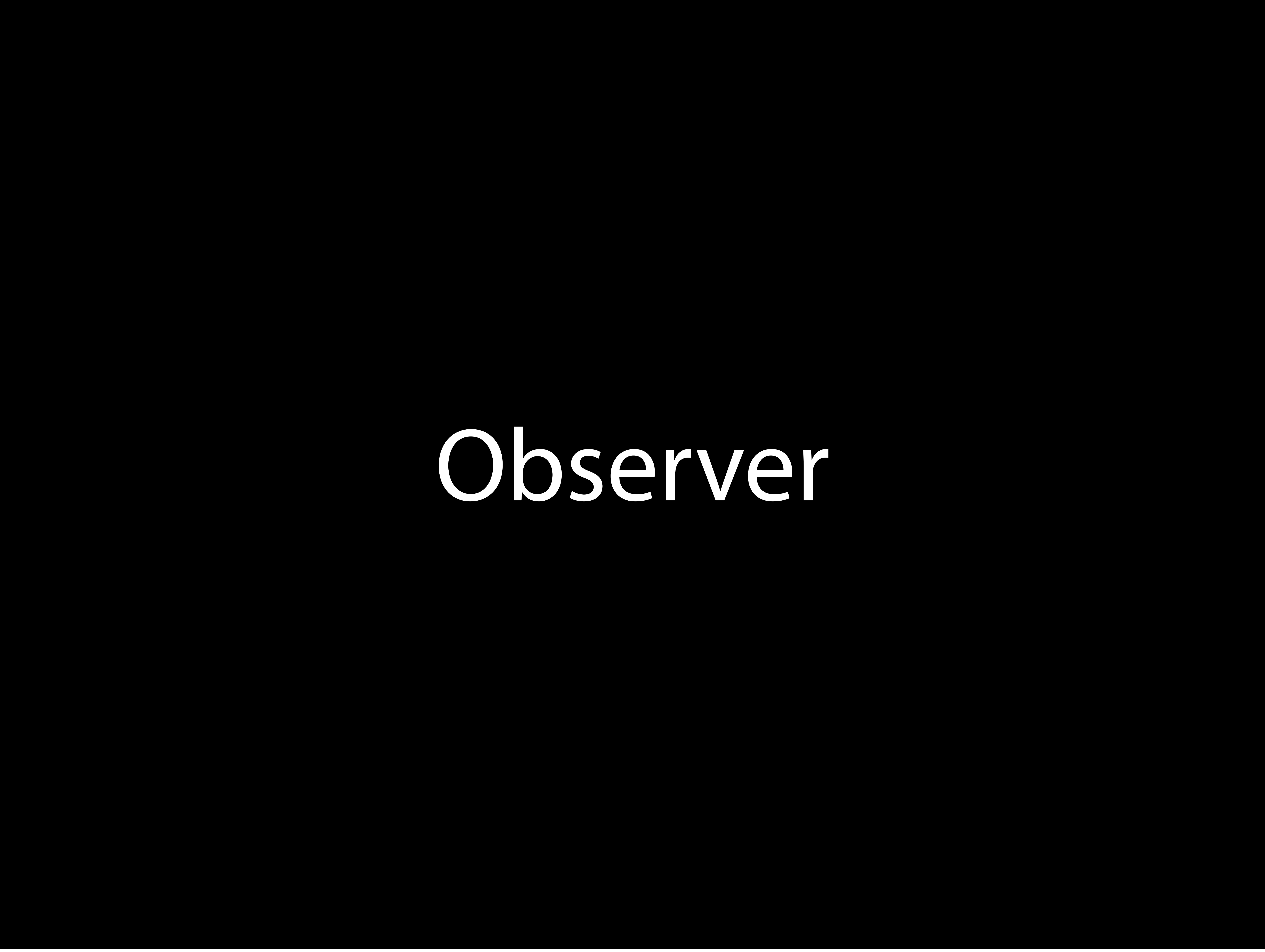
Wednesday, June 19, 13

Reachability
- One such pattern is to reconfigure your network/communication stack to behave differently based on different network service levels. Reconfigure your app to respond to the changes in network, on disconnection or in low bandwidth scenarios. Use a strategy pattern for the different network configurations, and the observer pattern to watch for the changes.

Strategy

Swap out Communication Strategies with the strategy pattern to capture the behavior in each special networking case. It could be that this is also returns canned responses that basically say the network is unusable.

You could also use the Null Object Pattern to tell your application that the network has been disconnected.


Observer

Wednesday, June 19, 13

In order to react to the changes in the network, we'd have to listen for changes to the network status. There are various ways to achieve this, but on iOS...

Reachability

On iOS we have a set Reachability APIs in the SystemConfiguration framework and various open source Objective C libraries of this to make it easier for us to use it.

Call-Callback

Wednesday, June 19, 13

Call-Callback pattern

A new hardware feature in mobile phones these days is the dual core CPU. This is only going to get more extreme as we reach the limits of Moore’s law and Physics itself. Concurrency is going to be more and more important to mobile apps, and should already be important as all your network operations should be performed in an asynchronous manner. The Call-Callback pattern is a great way to know when a particular unit of work is done, and from a different thread or worker queue.

```
- (void) someAsyncMethodCompletion:(void(^)
(id))block;
{
 dispatch_async(background_queue, ^{
      //...do work here, in the background

      if (block) { block(result); }
   });
}
```

```
[obj someAsyncMethodCompletion:^(id obj) {
  dispatch_async(dispatch_get_main_queue(), ^{

     //update your UI over here

  });
}];
```

Quiche

£3.50
£4.00
£4.30
£4.30
£4.30

farmhouse lunches

All served with French Bread, Pickles and Garnished with Salad

Your choice of our own cooked meats:

Ham	£6.50
Honey Roast Ham	£6.50
Beef	£6.50
Pork	£6.50
Turkey	£6.50
Quarter Chicken	£6.50
2 Sausage Rolls	£5.50
1 Pastie	£5.50
or:	
Cheddar	£5.30
Stilton	£5.30
Brie	£5.30

hot roast lunches

These dishes include New Potatoes, Cauliflower cheese, Carrots and Peas

Roast Pork with Crackling	£8.50
Rare Roast Beef	£8.50
Roast Lamb	£8.50
Roast Turkey Breast	£8.50

casseroles

All served with Jacket or New Potatoes

Venison and Boar	£9.00
Vicars	£9.00
Sweeney's	£9.00

Sweeney & Todd

hot pies

Please ask the Waitress for the Daily Selection from:

Steak	£6.50
Steak and Oyster	£6.50
Steak and Mushroom	£6.50
Steak and Kidney	£6.50
Minced Beef and Onion	£6.50
Yorkshire Ham and Stilton	£6.50
Chicken and Sweetcorn	£6.50
Lamb and Mint	£6.50
Pork and Apple	£6.50
Chicken, Honey and Mustard	£6.50
Venison and Boar	£6.50
Rump Steak and Stilton	£6.50
Chicken, Broccoli and Stilton	£6.50
Beef and Horseradish	£6.50
Chicken and Leek	£6.50
Cheese and Vegetable	£6.50
Five Nations	£6.50
Vicars	£6.50
Steak and Ale	£6.50
Chicken Cajun	£6.50
Sweeney's	£6.50
Turkey and Bacon	£6.50
Chicken and Ham	£6.50

PLUS VARIOUS OTHERS

Benefit: Menu

Selecting a solution to your design problem can be as simple as picking an item from a menu

Wednesday, June 19, 13

Benefit: Communication

As developers, having a common vocabulary to work with improves our communication effectiveness. We can use this vocabulary to come up with new solutions.

Wednesday, June 19, 13

Benefit: Speedier quality development

Why did rails become so popular? Because it was opinionated, and made some decisions for you every time. This time spent making decisions, especially the same ones, over and over again can slow you down without even realizing it.

It was important, his father said, to craft the backs of cabinets and fences properly, even though they were hidden.

Steve Jobs by Walter Isaacson

“He loved doing things right. He even cared about the look of the parts you couldn’t see.”

- Steve Jobs

Steve Jobs by Walter Isaacson, Chapter 1

It is important to care about the internal quality of products you build, as well as the external.

Design Patterns are proven techniques to add care and quality to your apps while not compromising quality in a given amount of development time.

saul@magicalpanda.com