

MODEL-VIEW-CONTROLLER AND TABLEVIEWS

Nick Chen
Fall 2015
talentspark.io

PREVIOUSLY

The screenshot shows the Xcode IDE with two main panes. The left pane displays the `ViewController.swift` file, which contains Swift code for a view controller. The right pane shows the `Main.storyboard` interface, which includes a globe icon, a button labeled "Click me", and a text field with placeholder text.

```
// View Controller.swift
// GitHubDemo
//
// Created by Nick Chen on 7/28/15.
// Copyright © 2015 TalentSpark. All rights reserved.

import UIKit

class ViewController: UIViewController {

    @IBOutlet weak var hello: UILabel!
    @IBOutlet weak var button: UIButton!
    @IBOutlet weak var textField: UITextField!

    @IBAction func buttonClicked(sender: UIButton) {
        hello.text = "Hello, Clicker!"
    }

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

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

Running GitHubDemo on iPhone 6

Main.storyboard (Base)

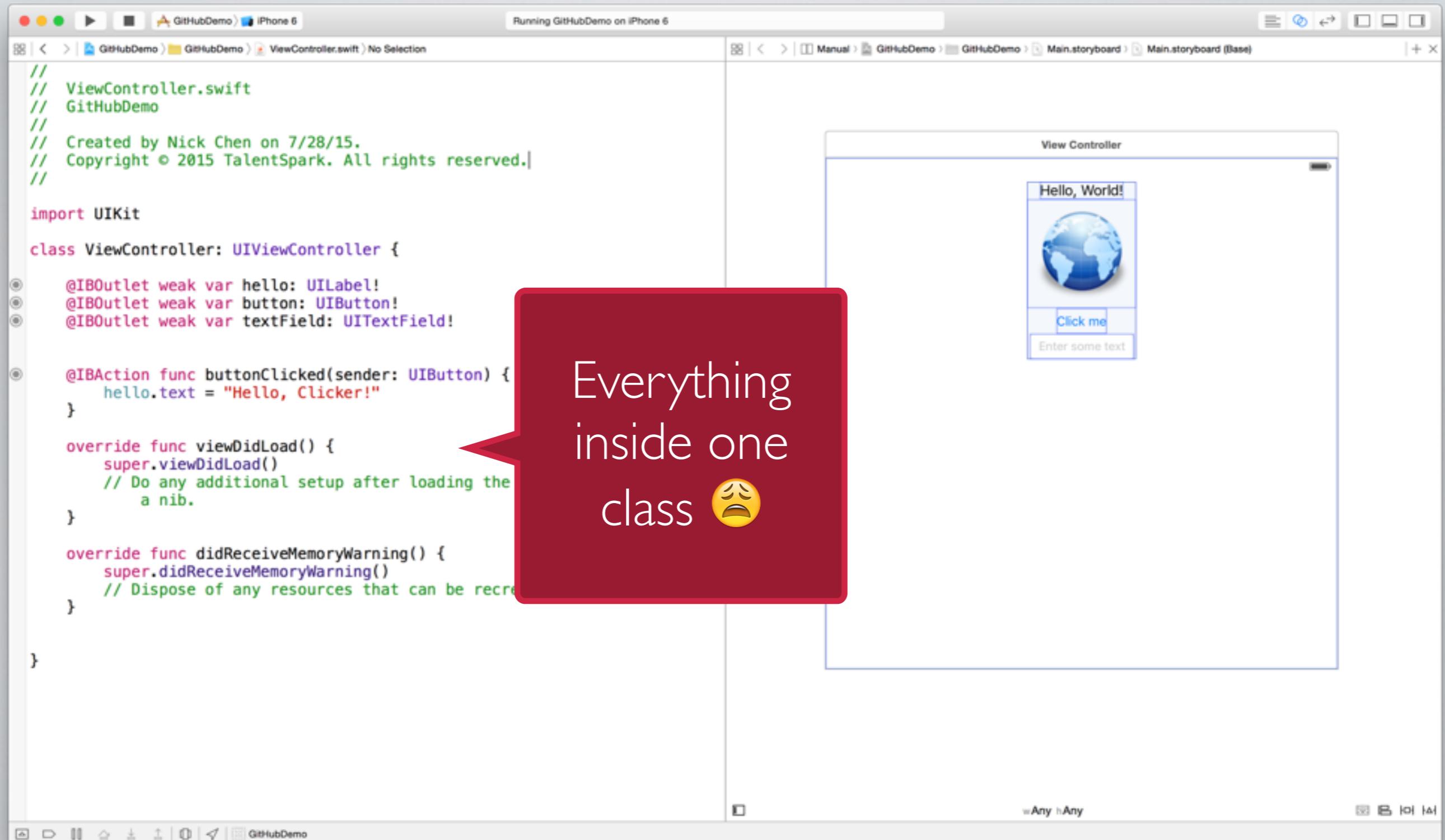
View Controller

Hello, World!

Click me

Enter some text

PREVIOUSLY



The screenshot shows the Xcode IDE with two main panes. The left pane displays the `ViewController.swift` file, which contains Swift code for a view controller. The right pane shows the storyboard interface for the `Main.storyboard` file, which includes a globe image, a button labeled "Click me", and a text field labeled "Enter some text". A red callout bubble points from the text "Everything inside one class 😞" to the `ViewController` class definition in the code.

```
// ViewController.swift
// GitHubDemo
//
// Created by Nick Chen on 7/28/15.
// Copyright © 2015 TalentSpark. All rights reserved.

import UIKit

class ViewController: UIViewController {

    @IBOutlet weak var hello: UILabel!
    @IBOutlet weak var button: UIButton!
    @IBOutlet weak var textField: UITextField!

    @IBAction func buttonClicked(sender: UIButton) {
        hello.text = "Hello, Clicker!"
    }

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the nib.
    }

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

Everything inside one class 😞

View Controller

Hello, World!

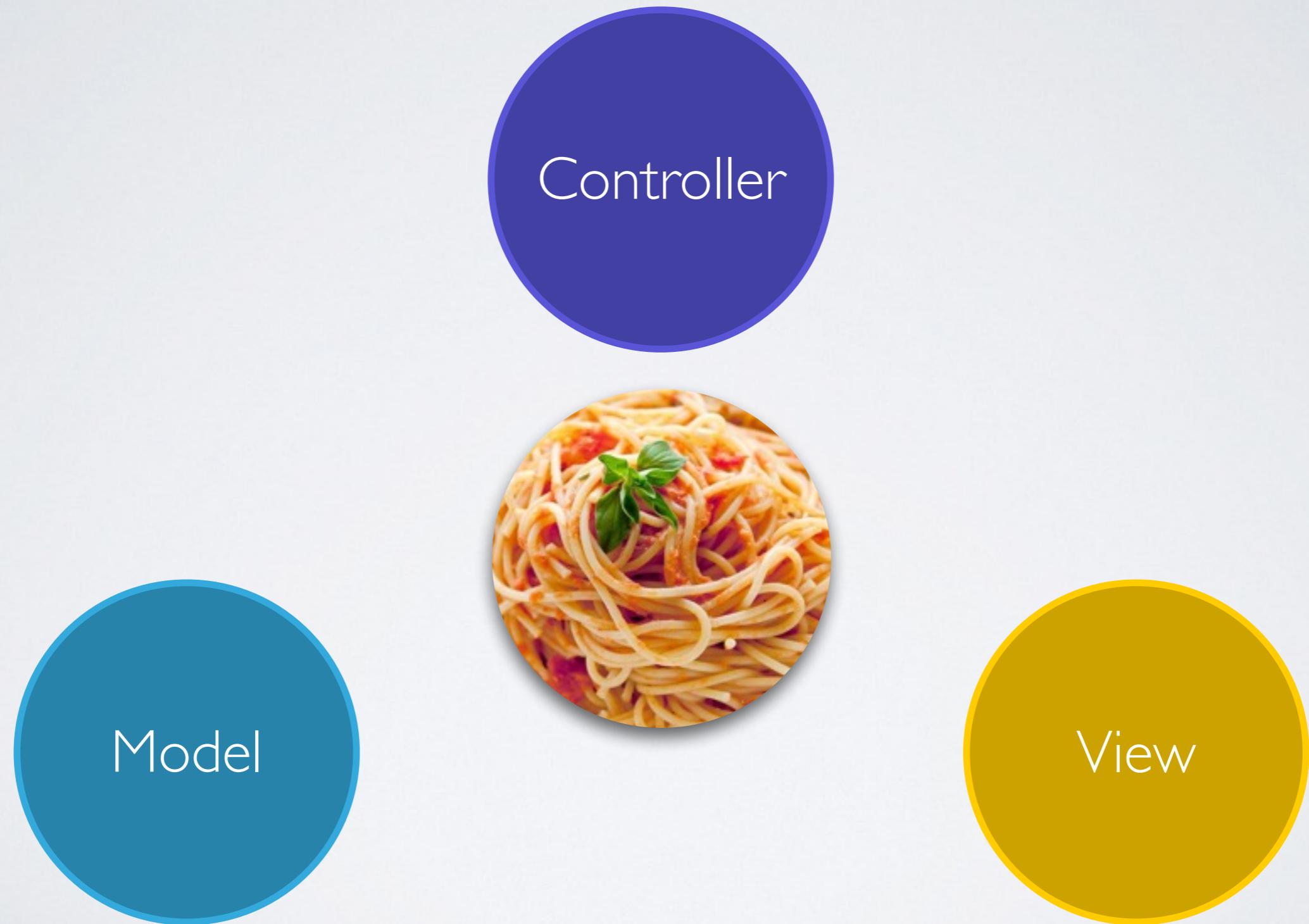
Click me

Enter some text

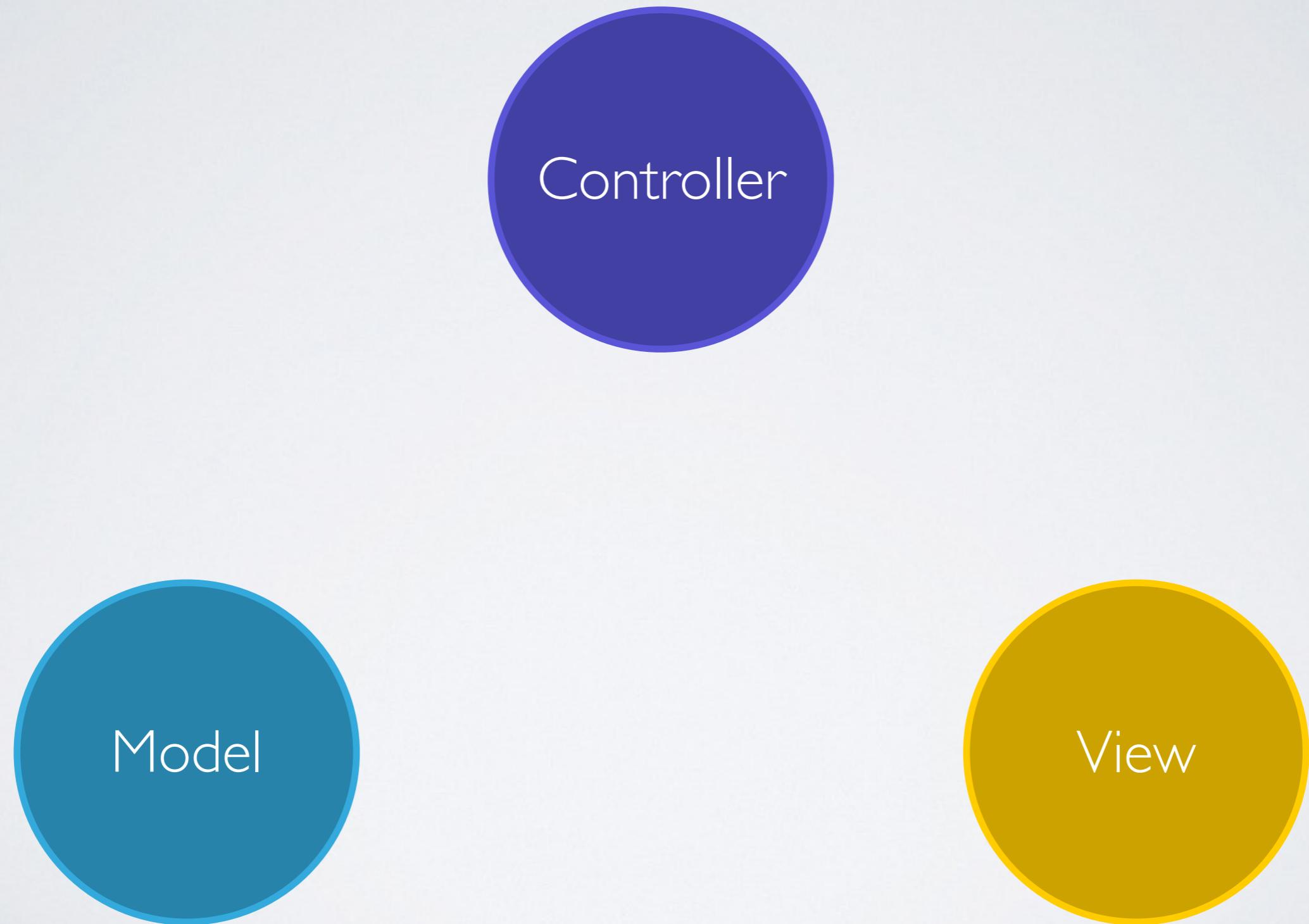
SPAGHETTI CODE



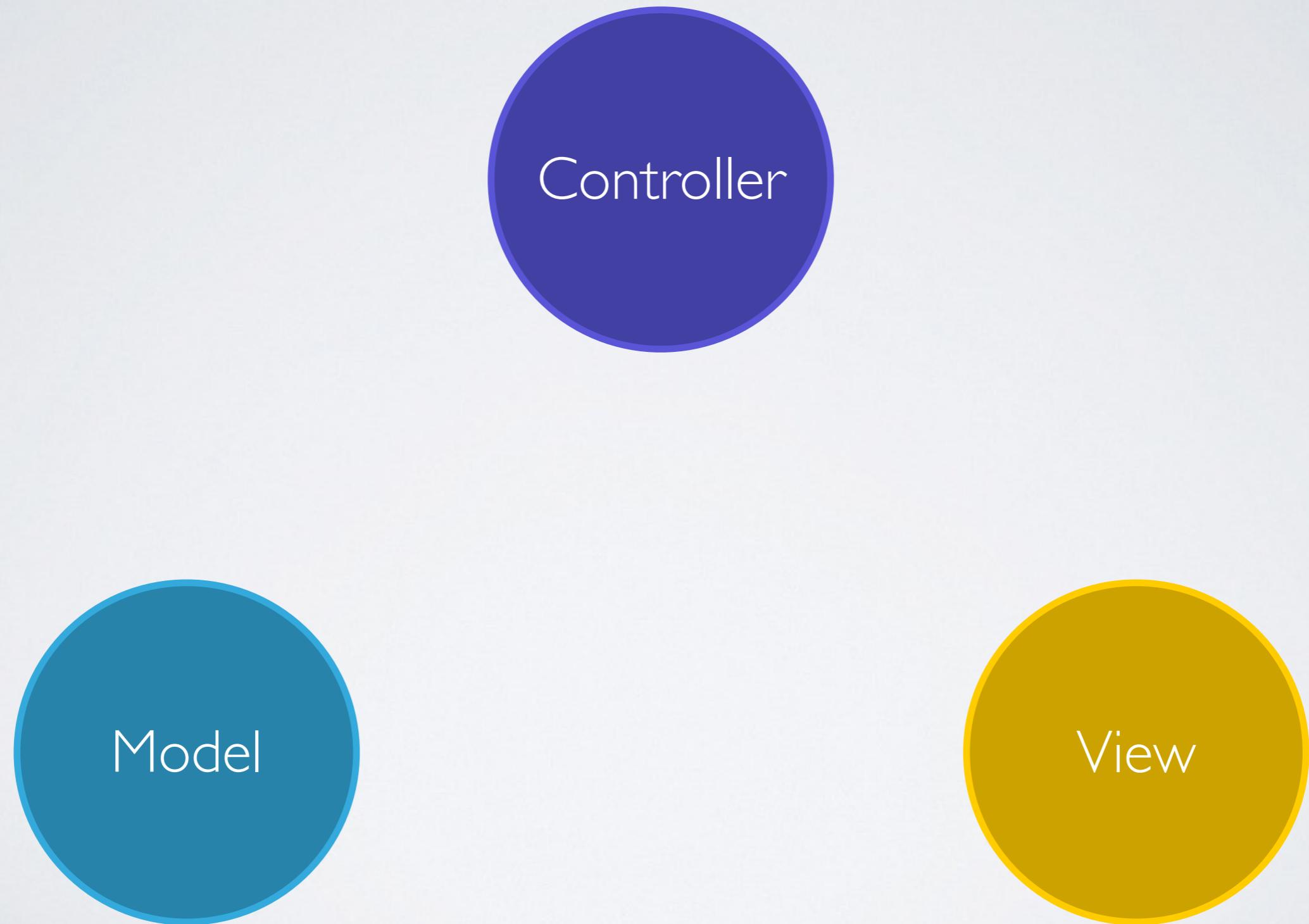
SPAGHETTI CODE



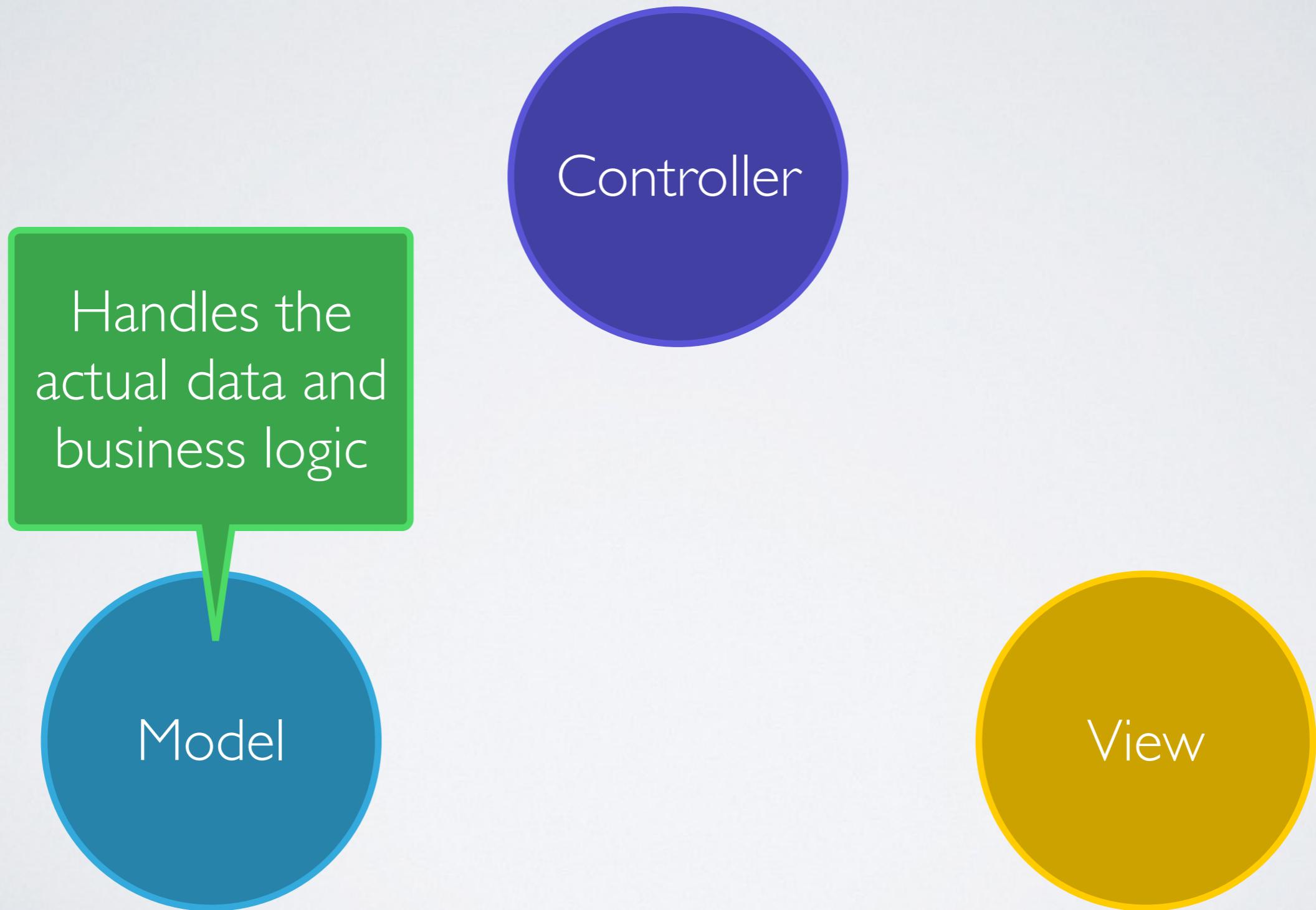
SPAGHETTI CODE



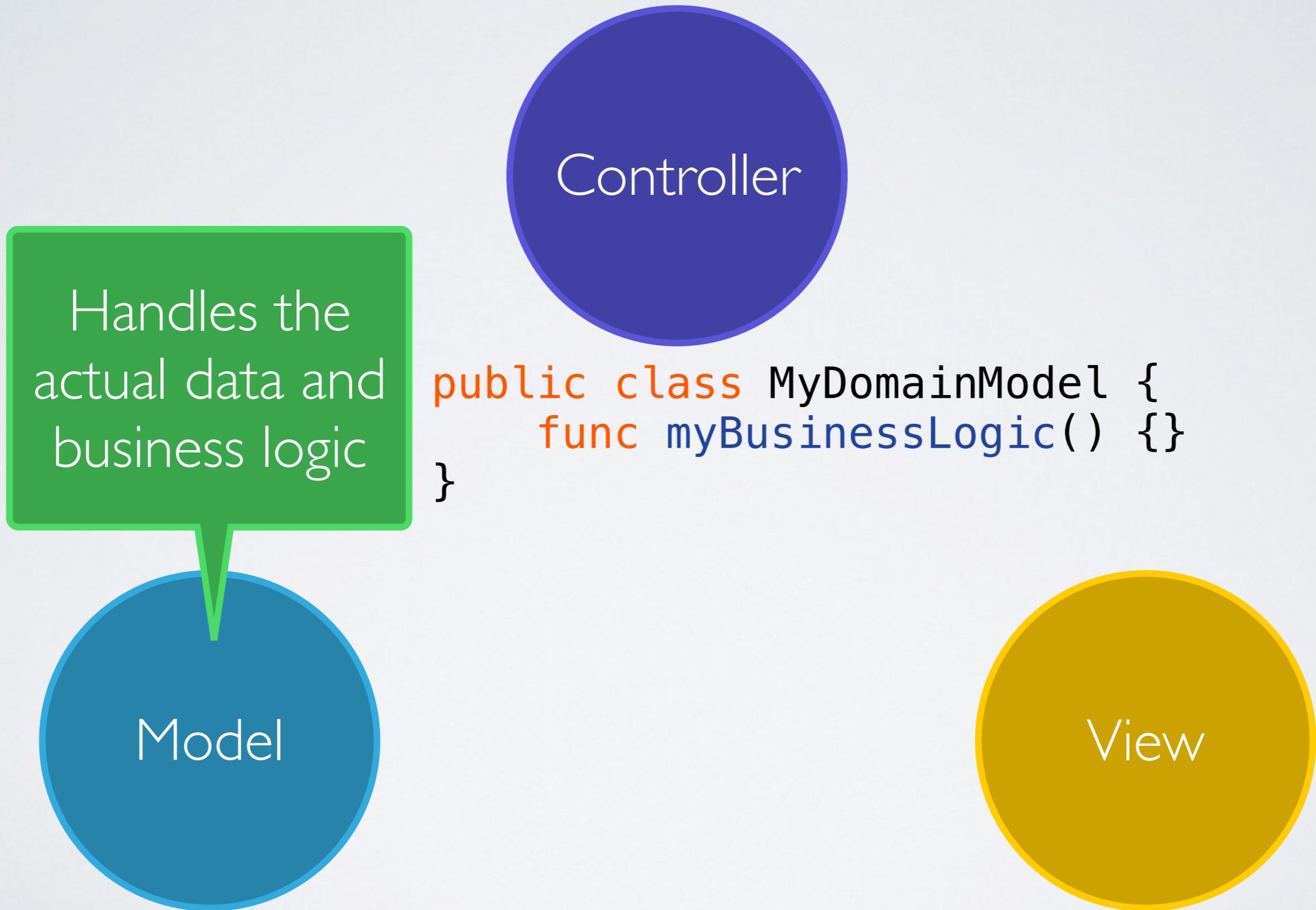
NON-SPAGHETTI CODE



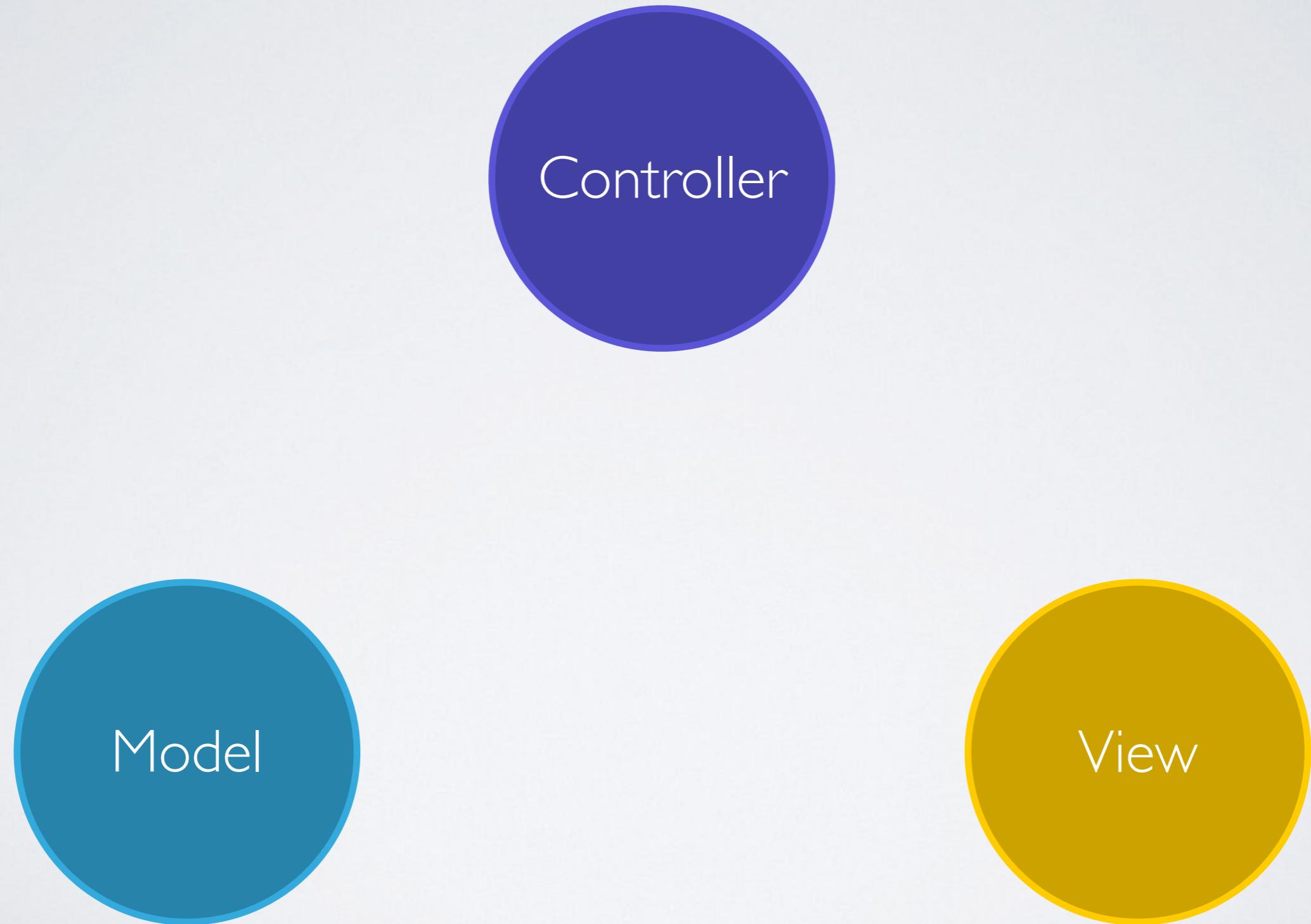
NON-SPAGHETTI CODE



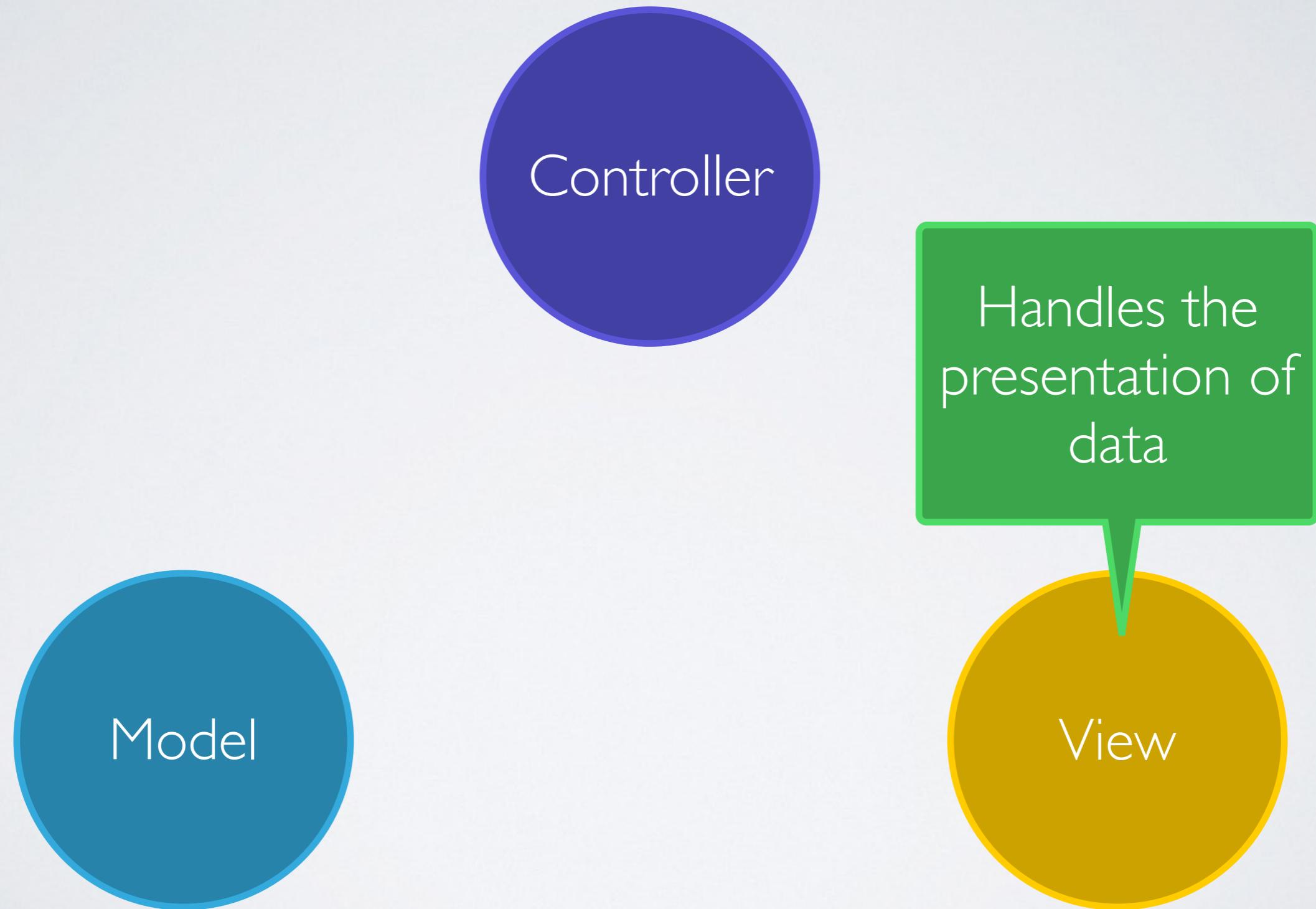
NON-SPAGHETTI CODE



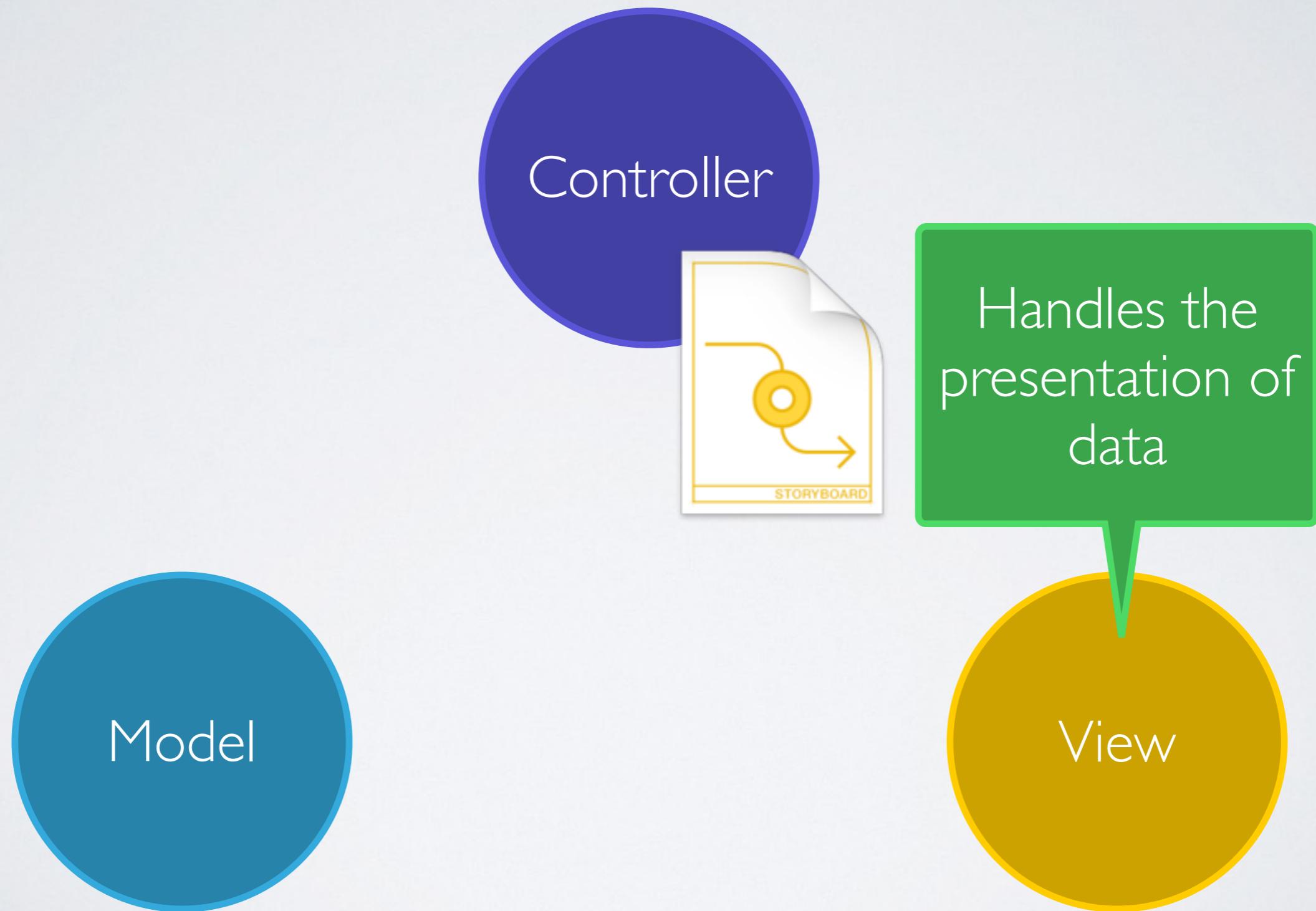
NON-SPAGHETTI CODE



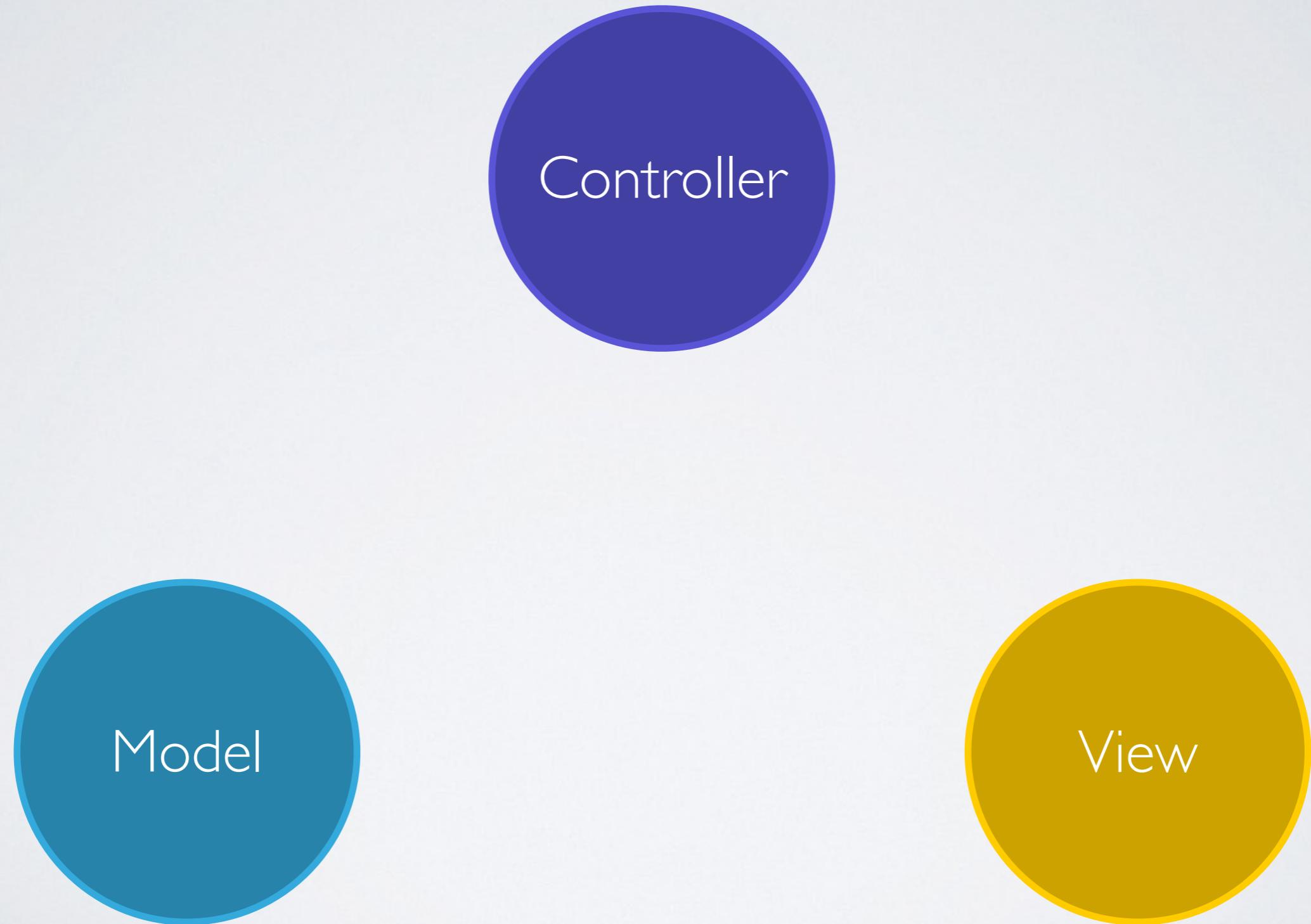
NON-SPAGHETTI CODE



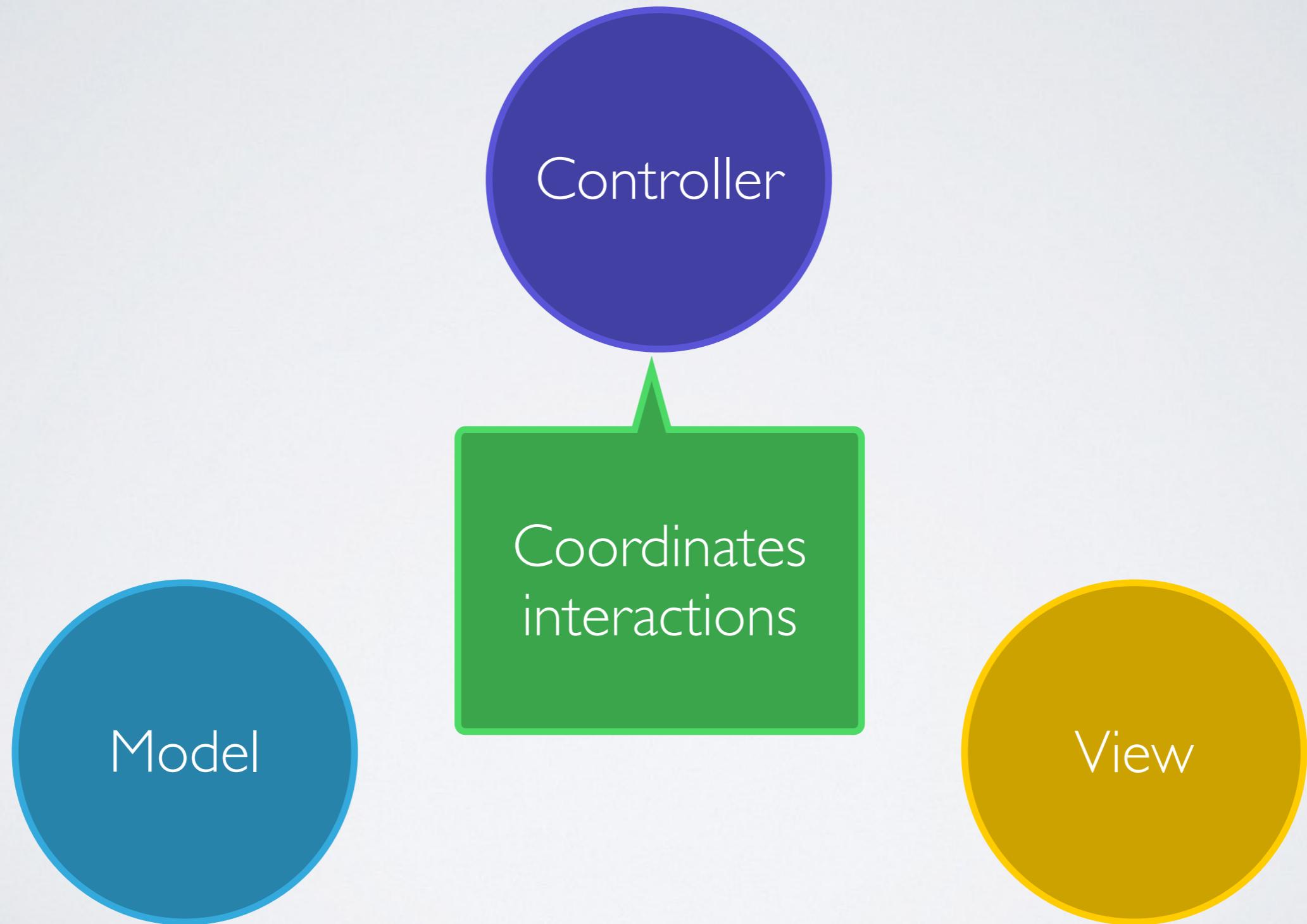
NON-SPAGHETTI CODE



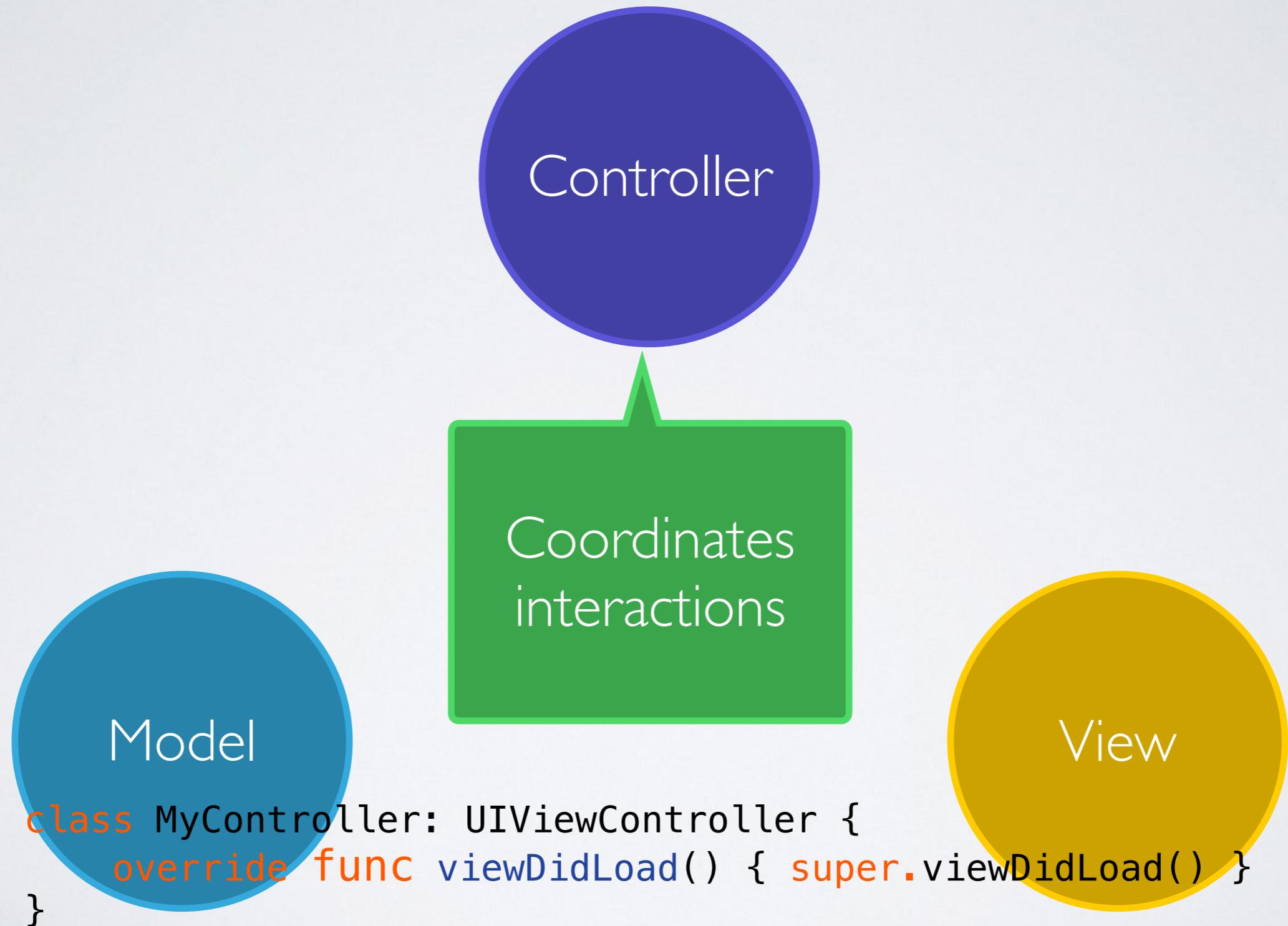
NON-SPAGHETTI CODE



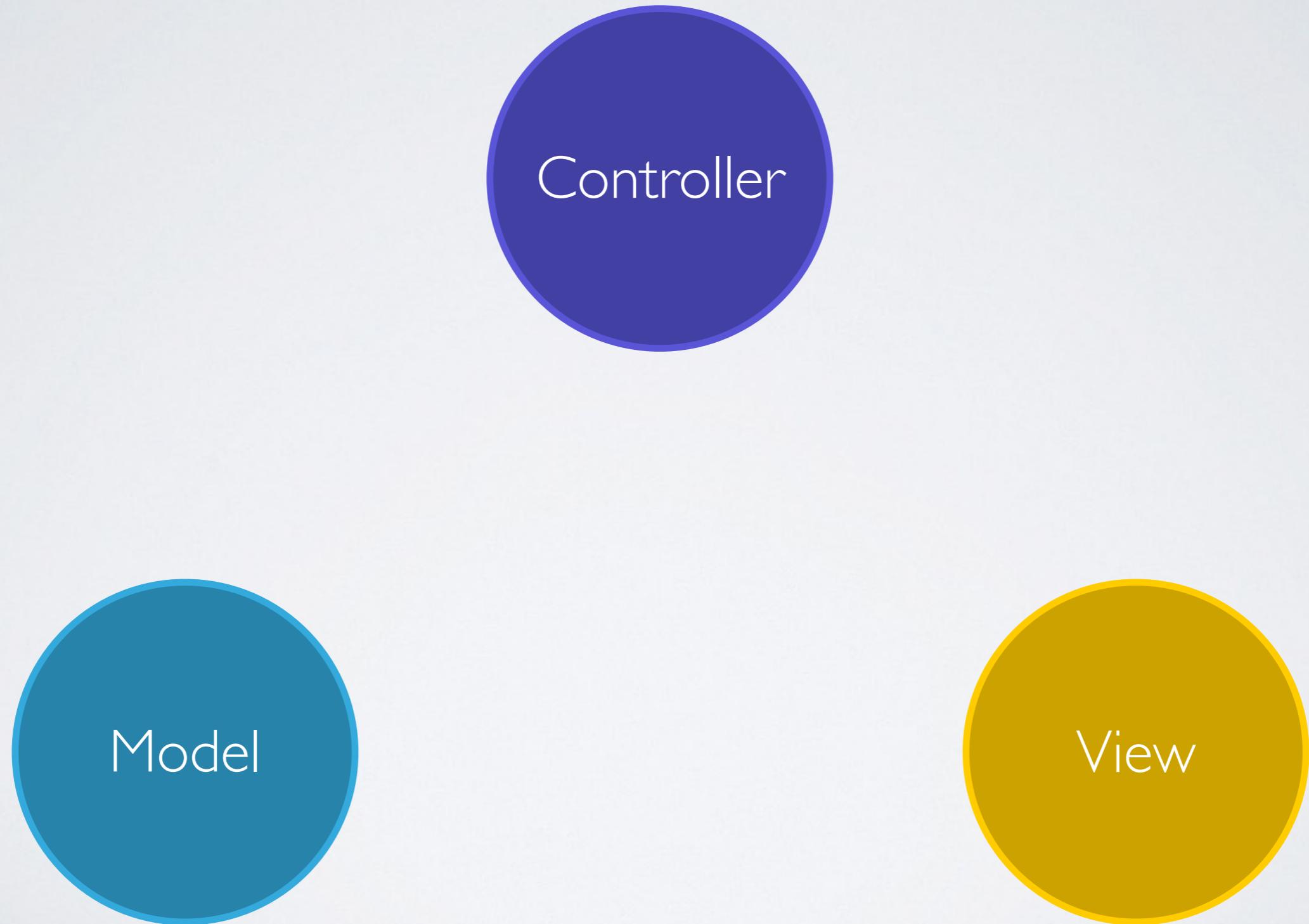
NON-SPAGHETTI CODE



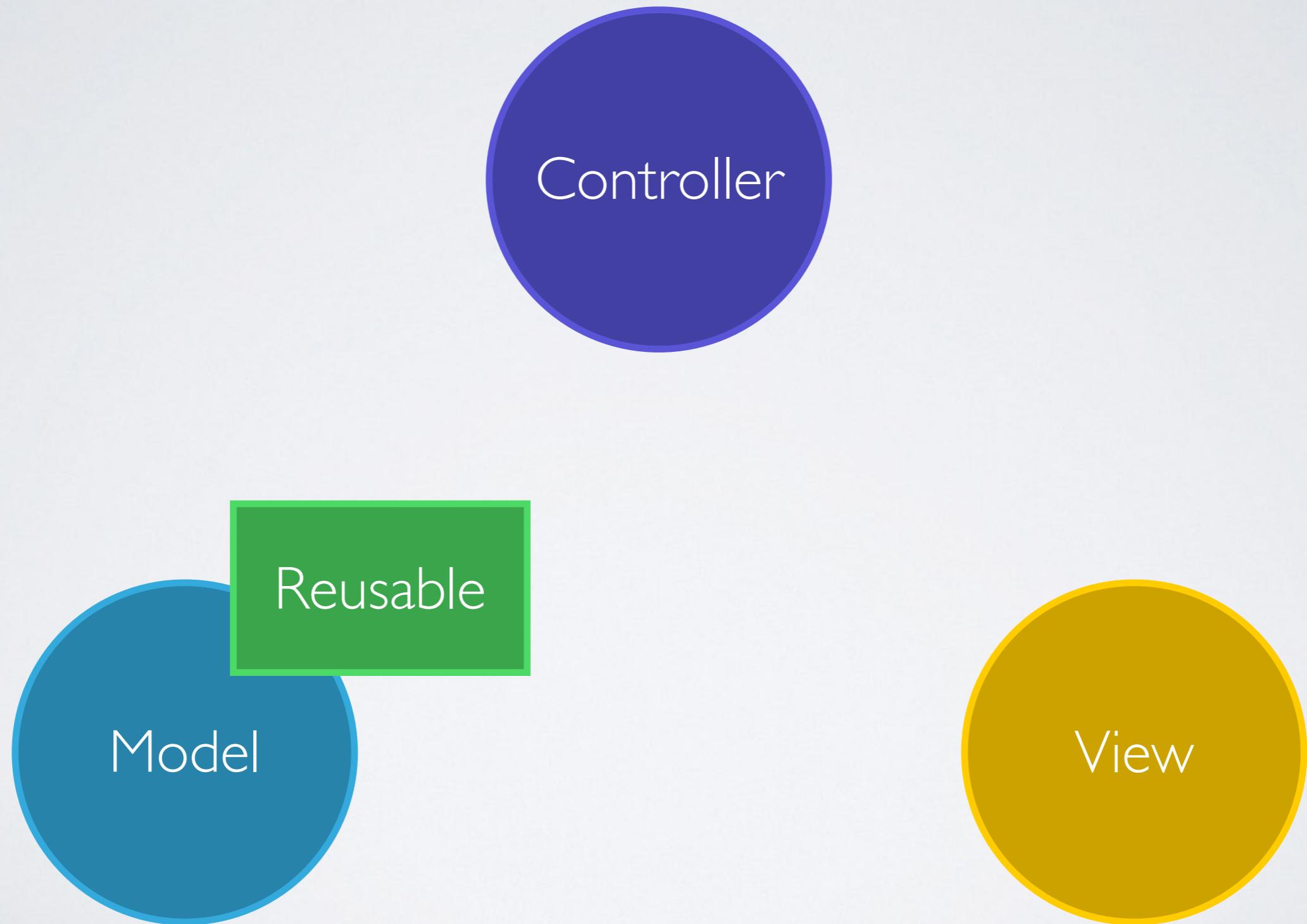
NON-SPAGHETTI CODE



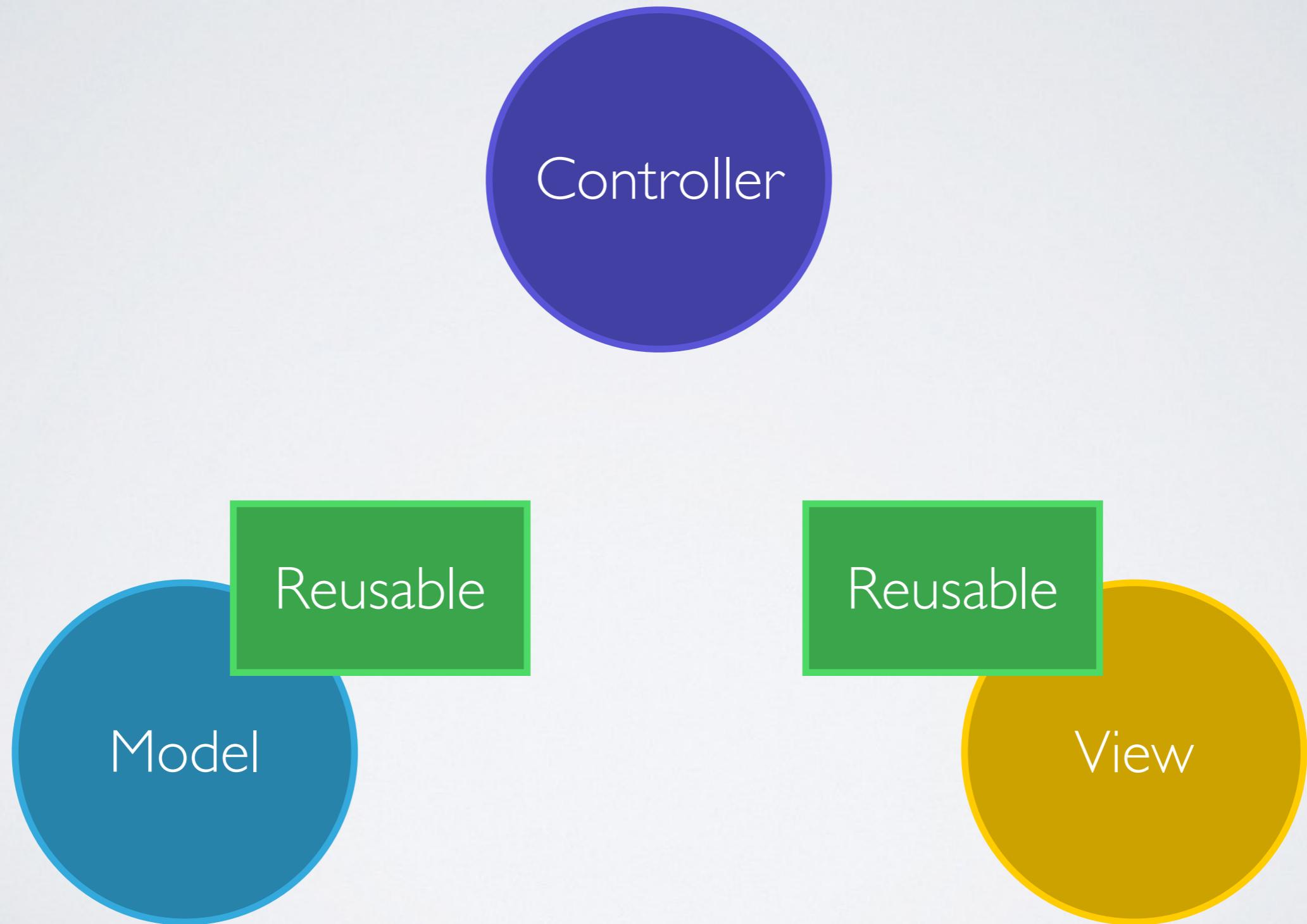
NON-SPAGHETTI CODE



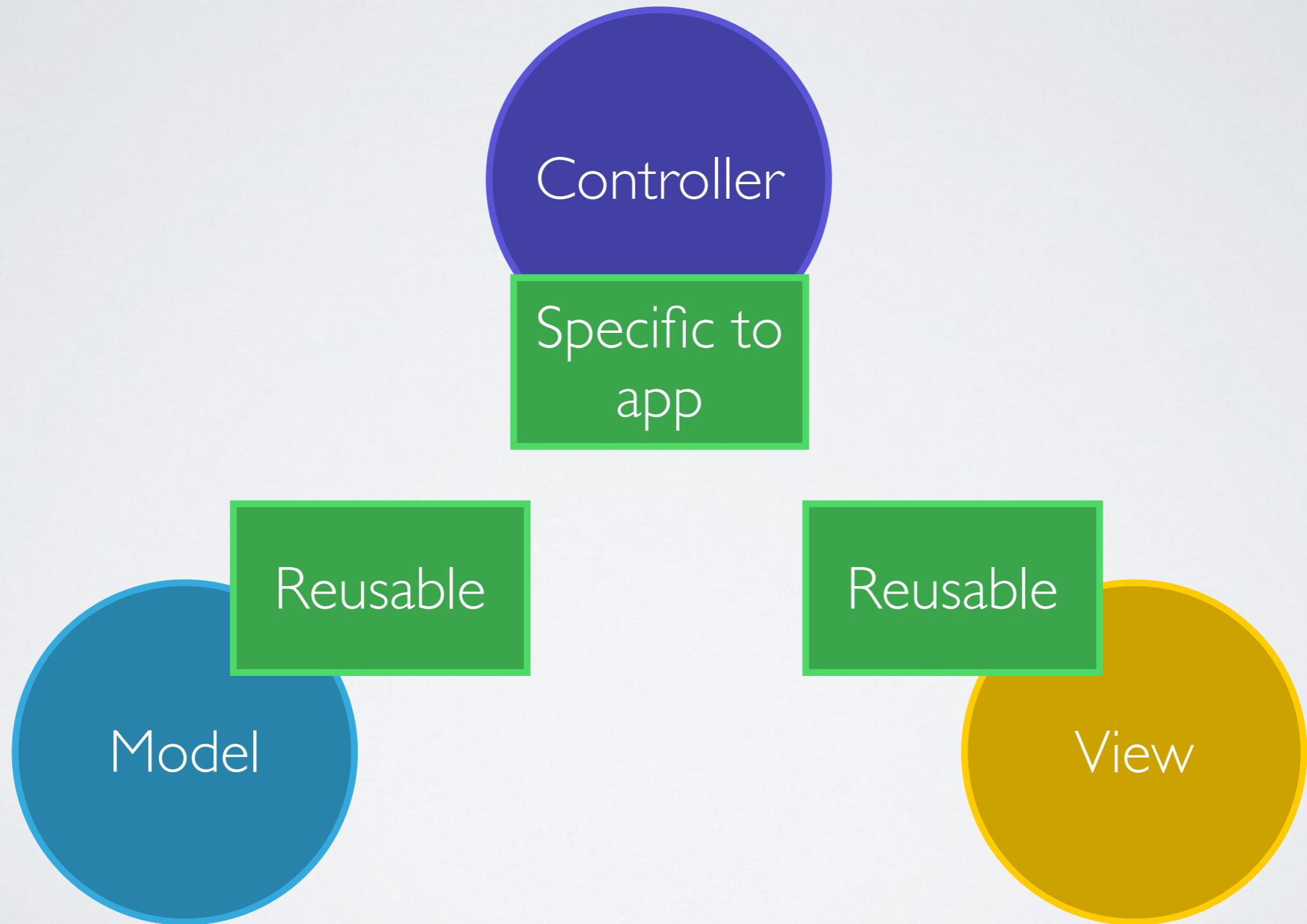
NON-SPAGHETTI CODE



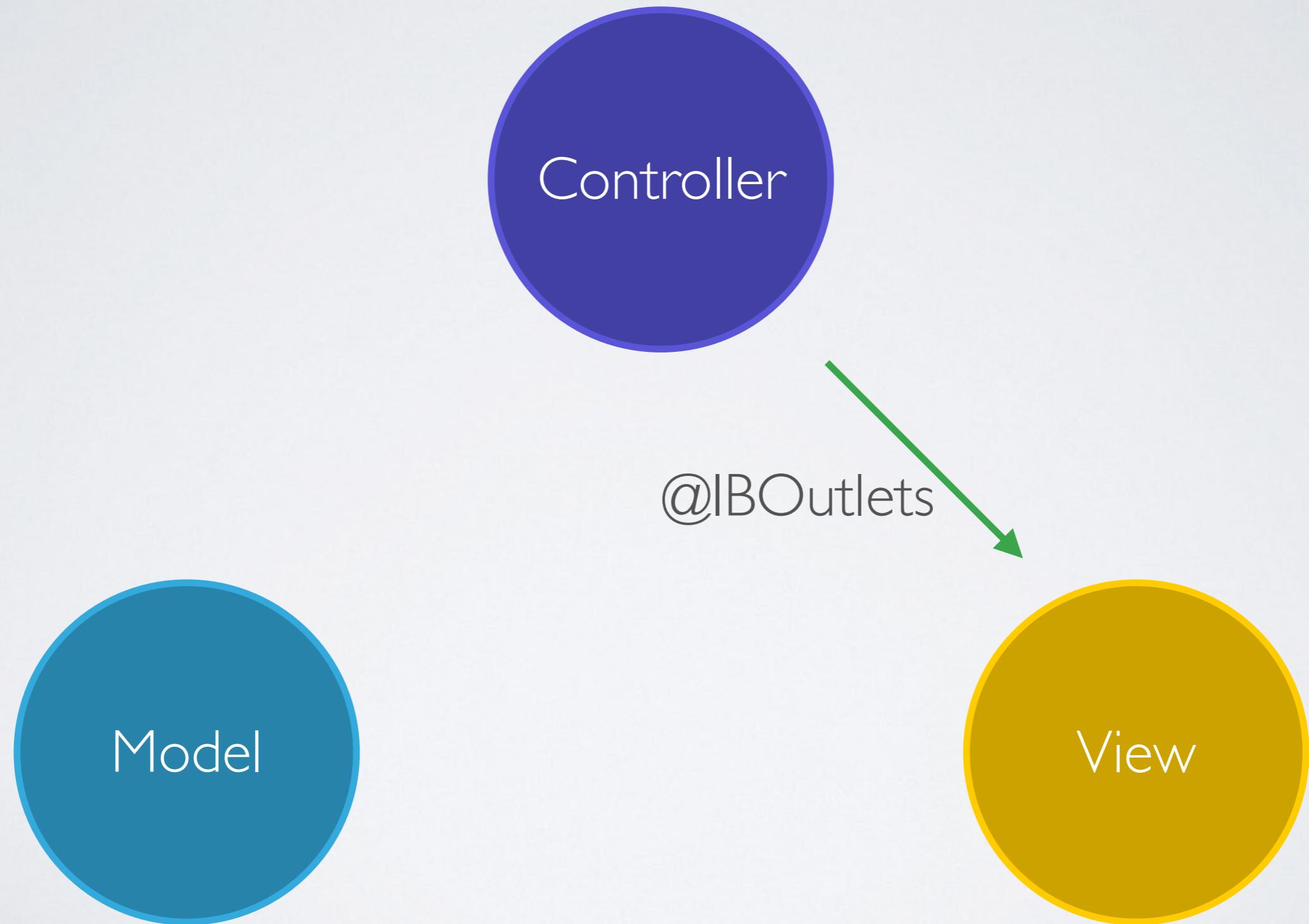
NON-SPAGHETTI CODE



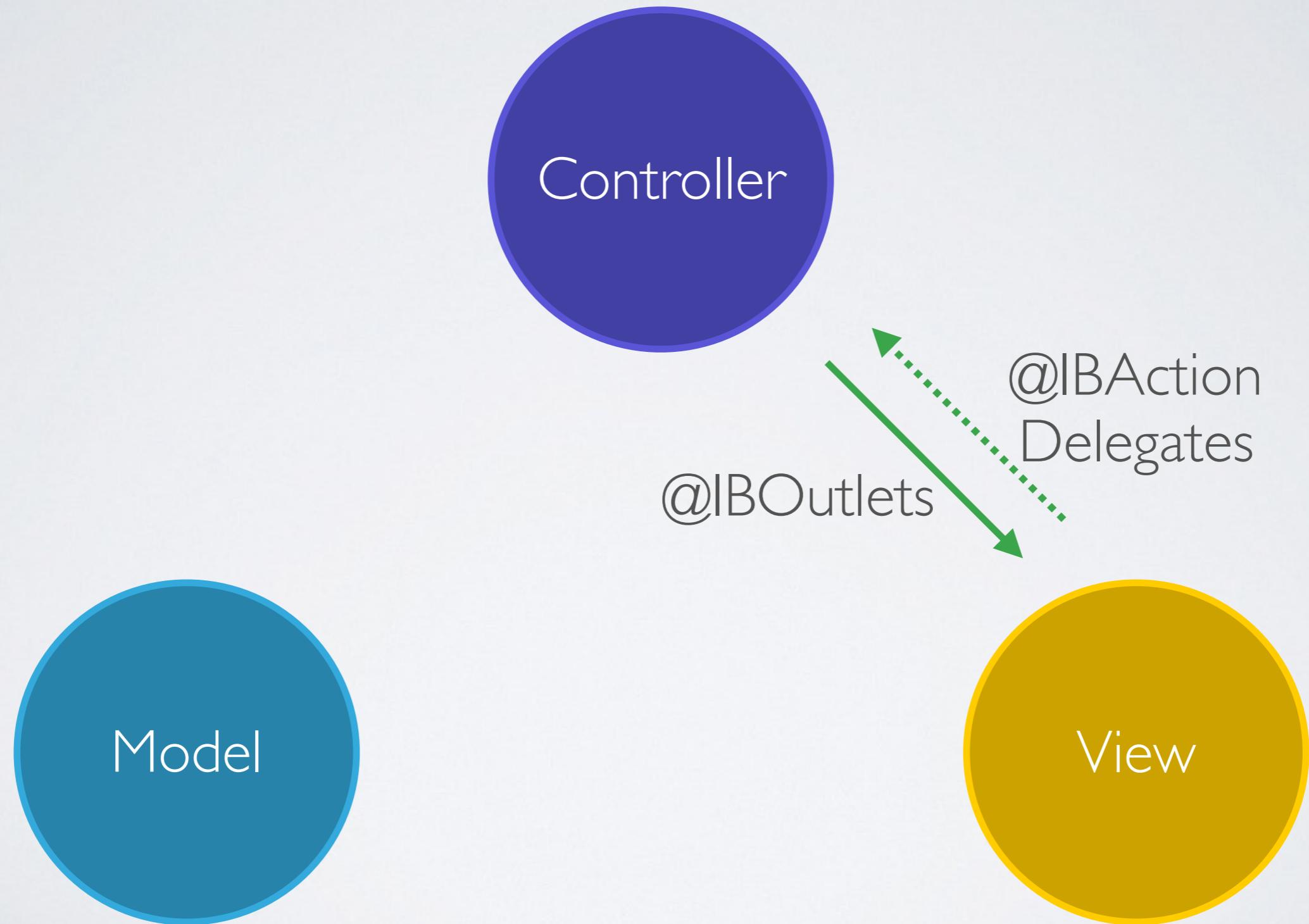
NON-SPAGHETTI CODE



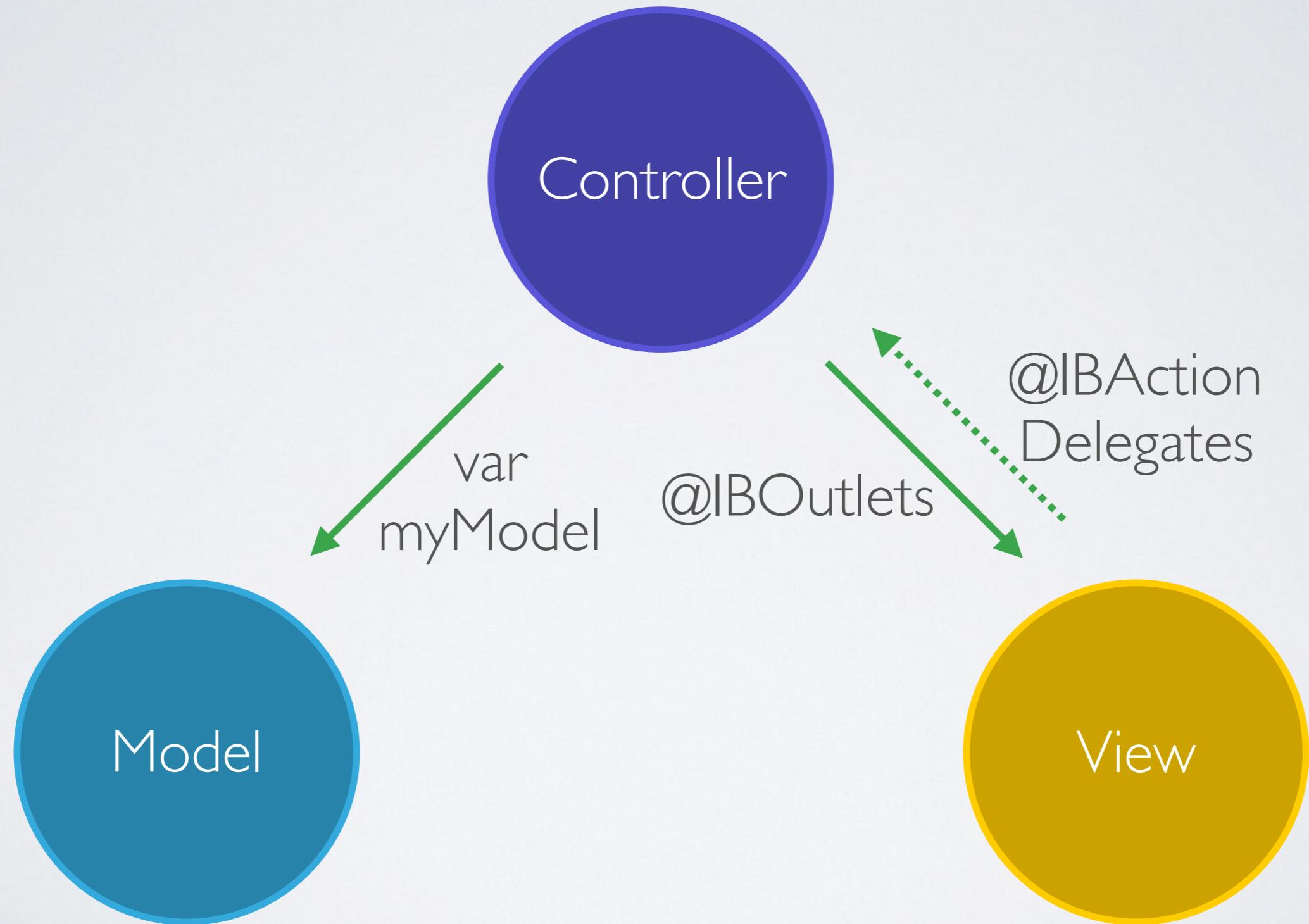
NON-SPAGHETTI CODE



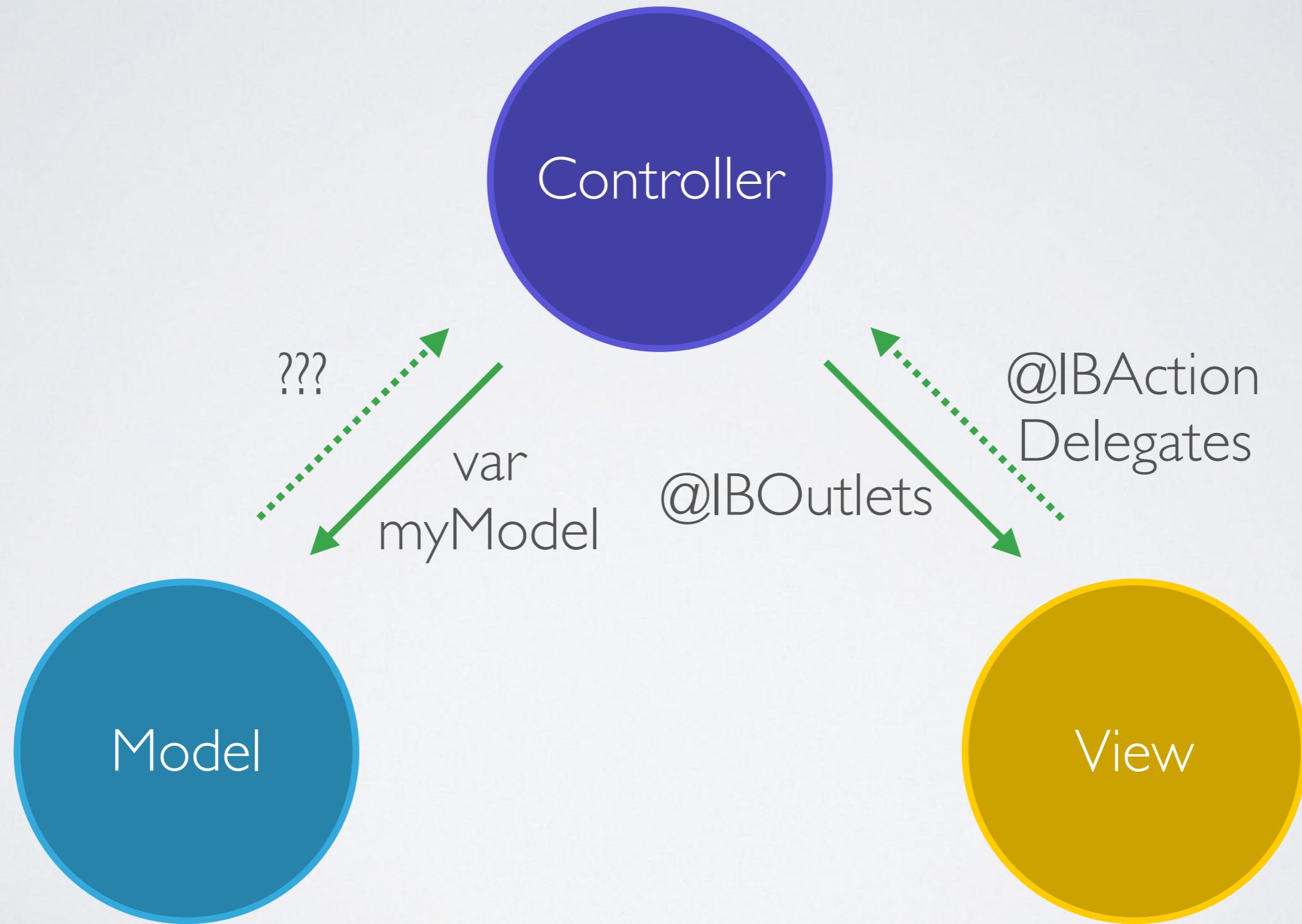
NON-SPAGHETTI CODE



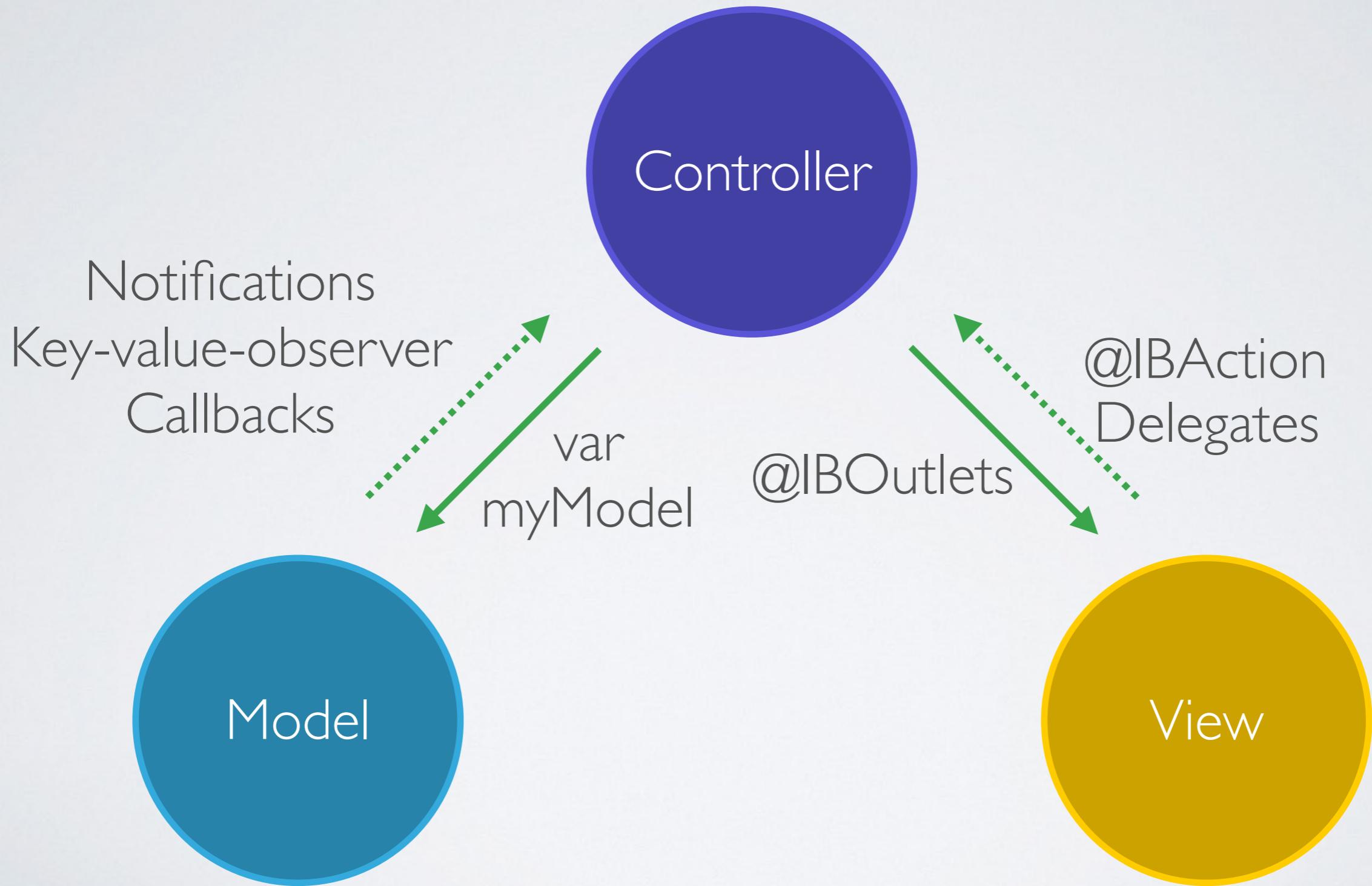
NON-SPAGHETTI CODE



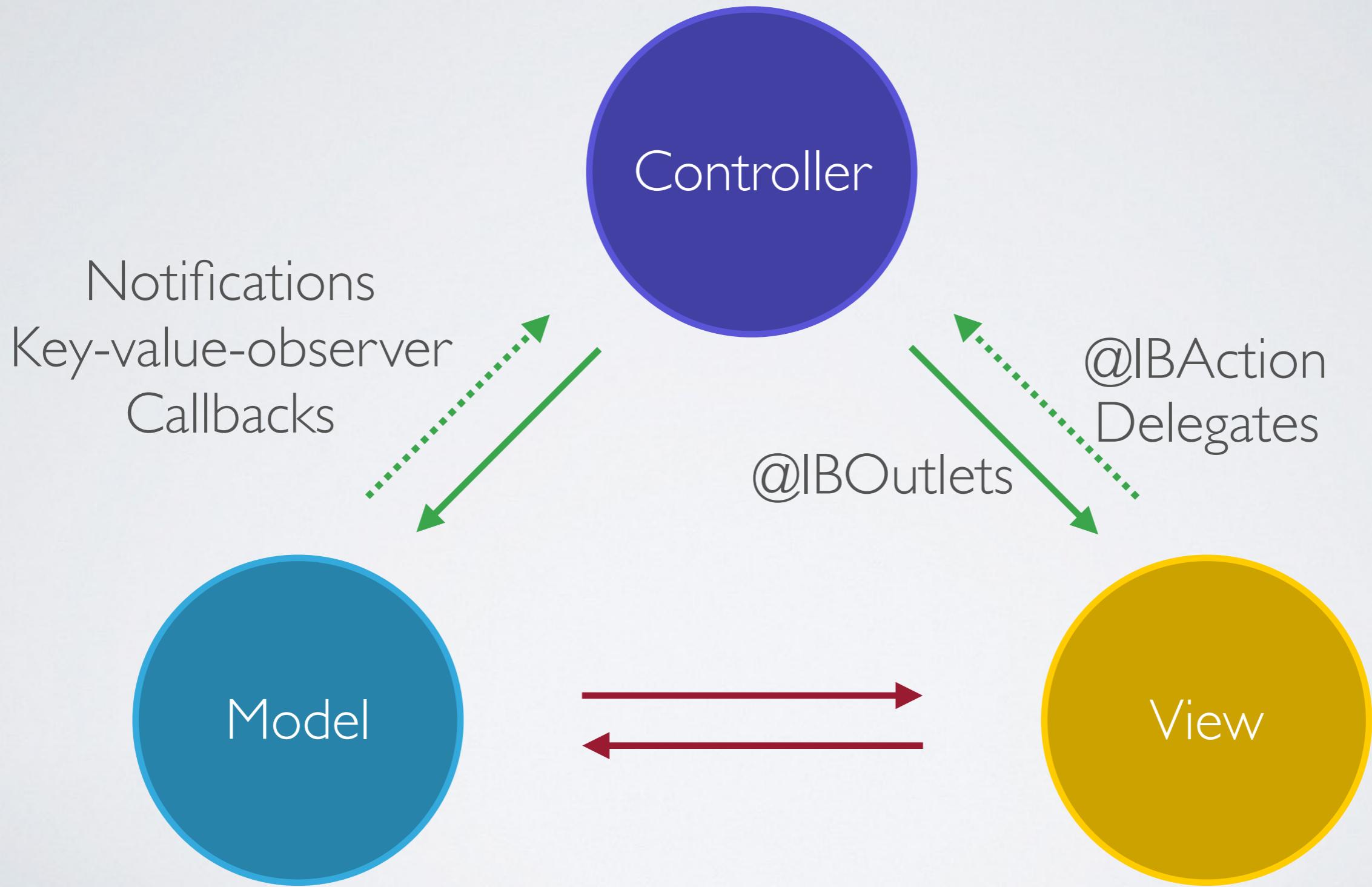
NON-SPAGHETTI CODE



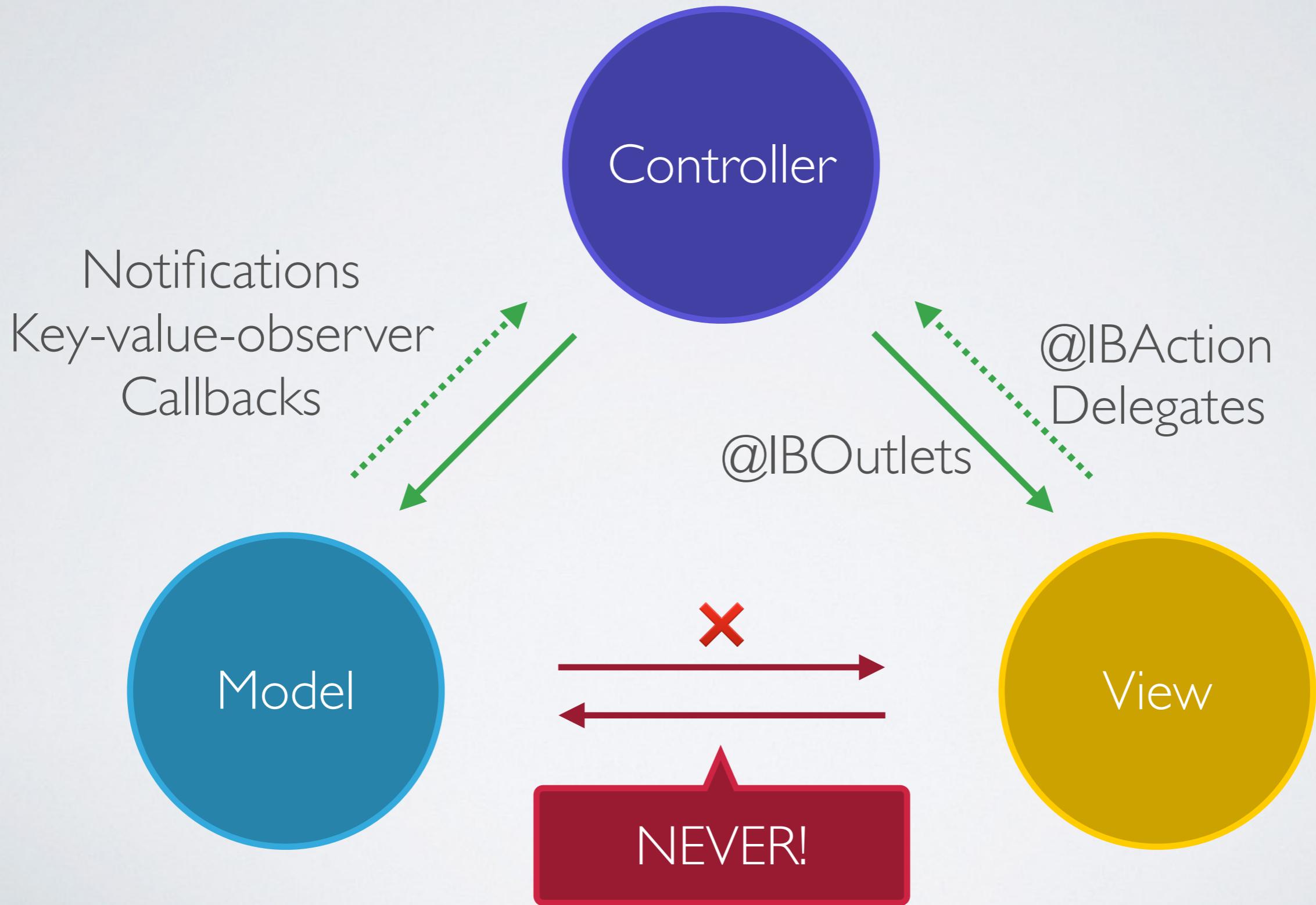
NON-SPAGHETTI CODE



NON-SPAGHETTI CODE

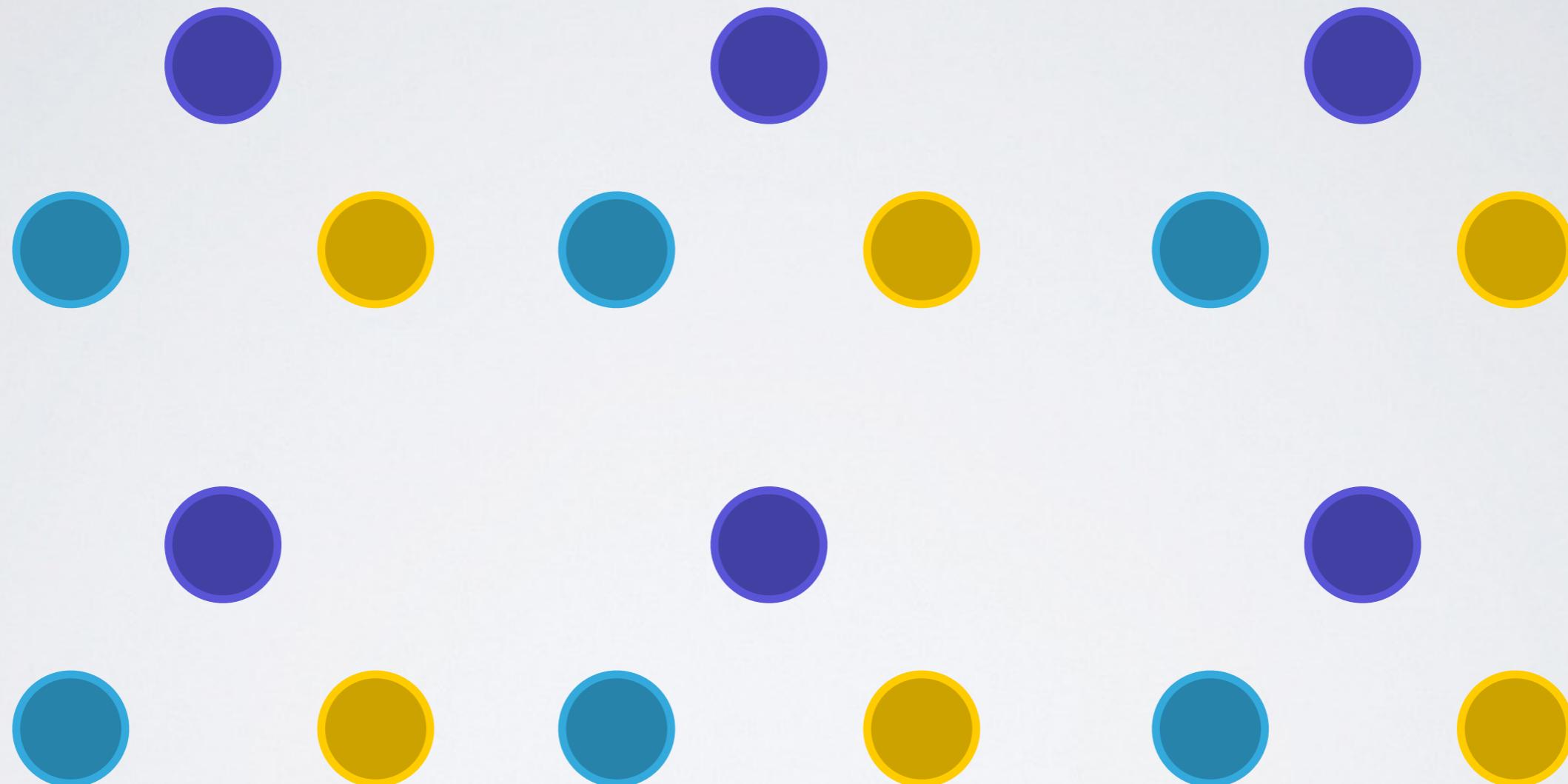


NON-SPAGHETTI CODE

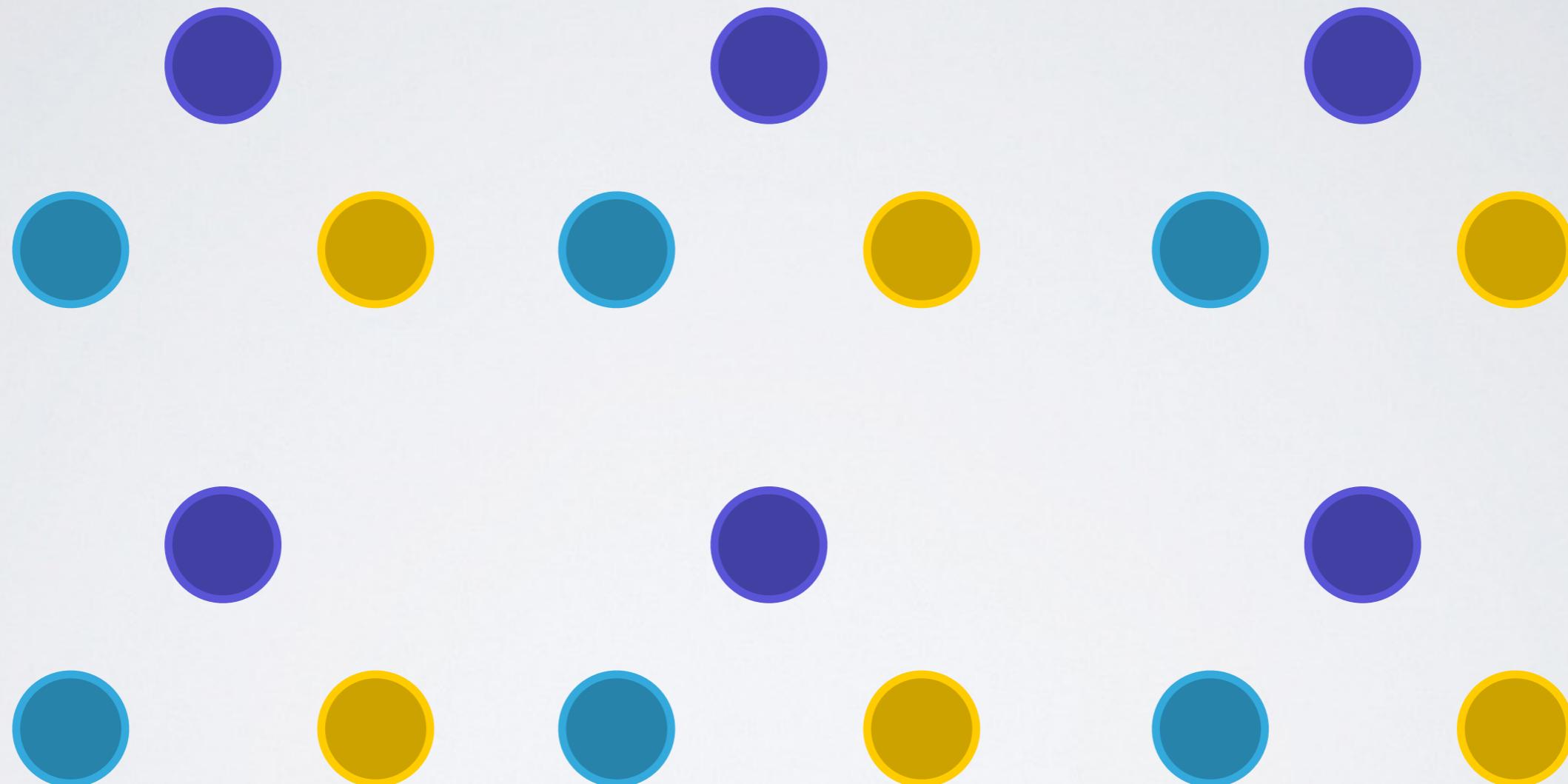


MULTIPLE MVCS

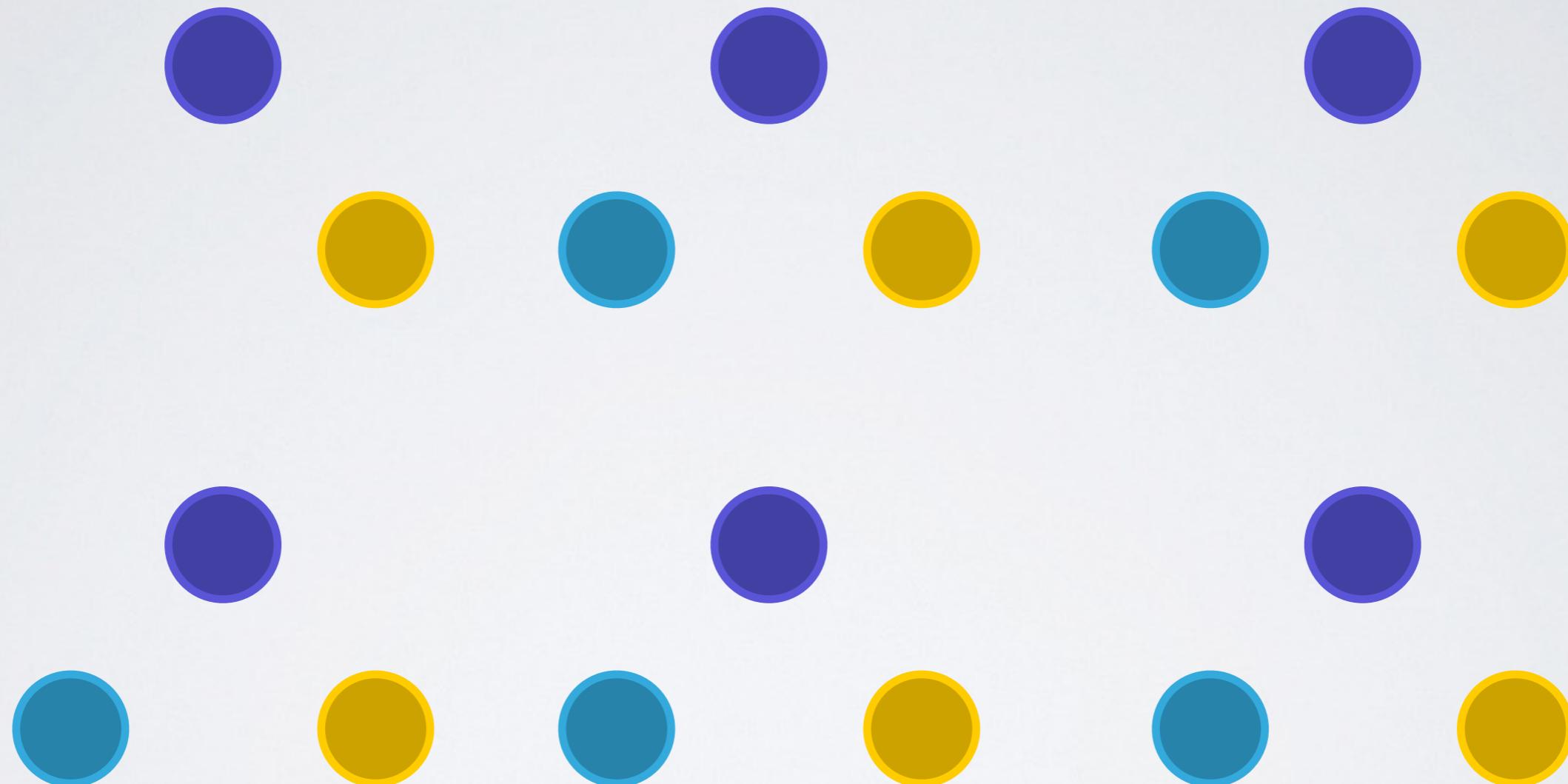
MULTIPLE MVCS



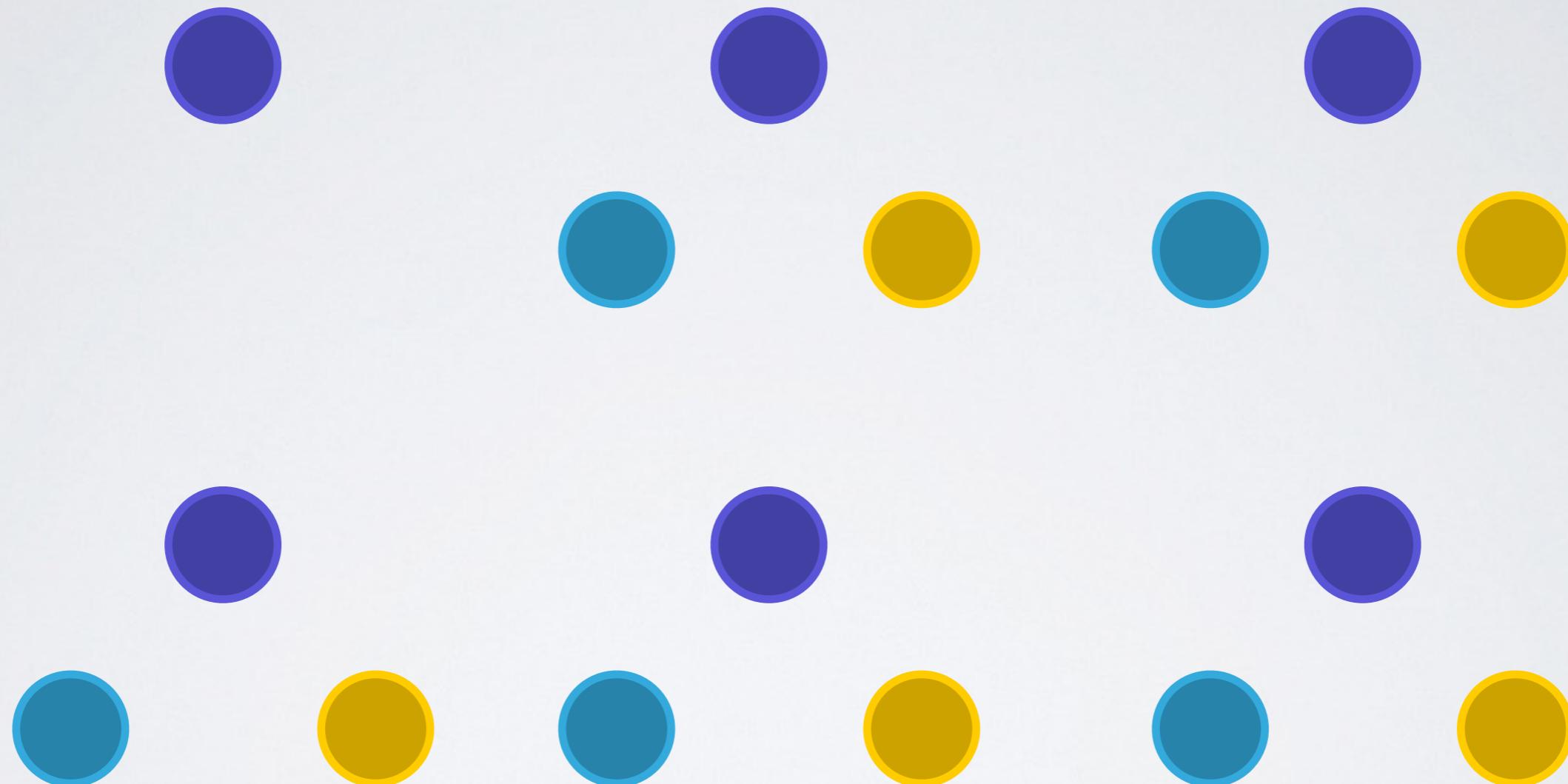
MULTIPLE MVCS



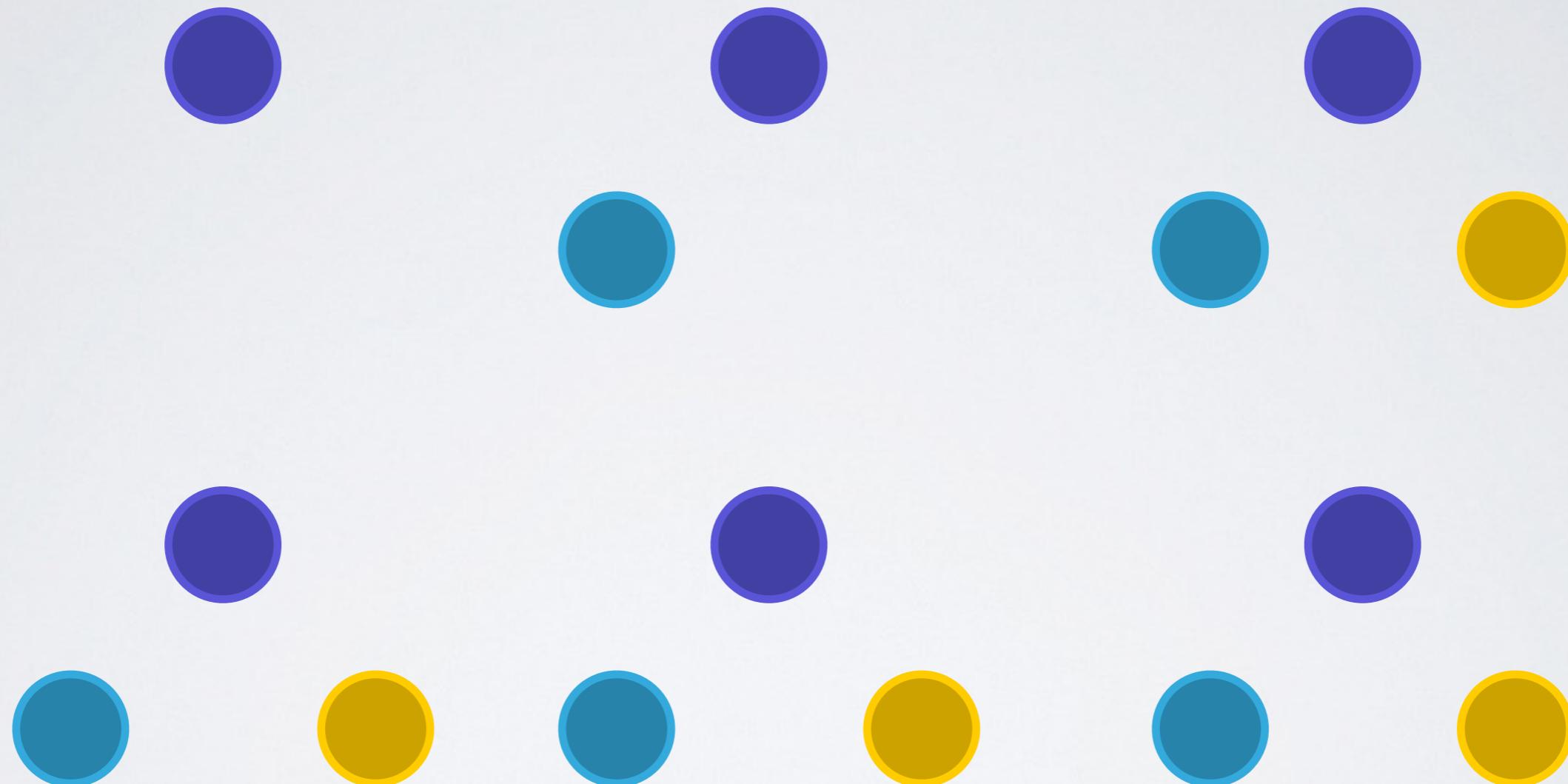
MULTIPLE MVCS



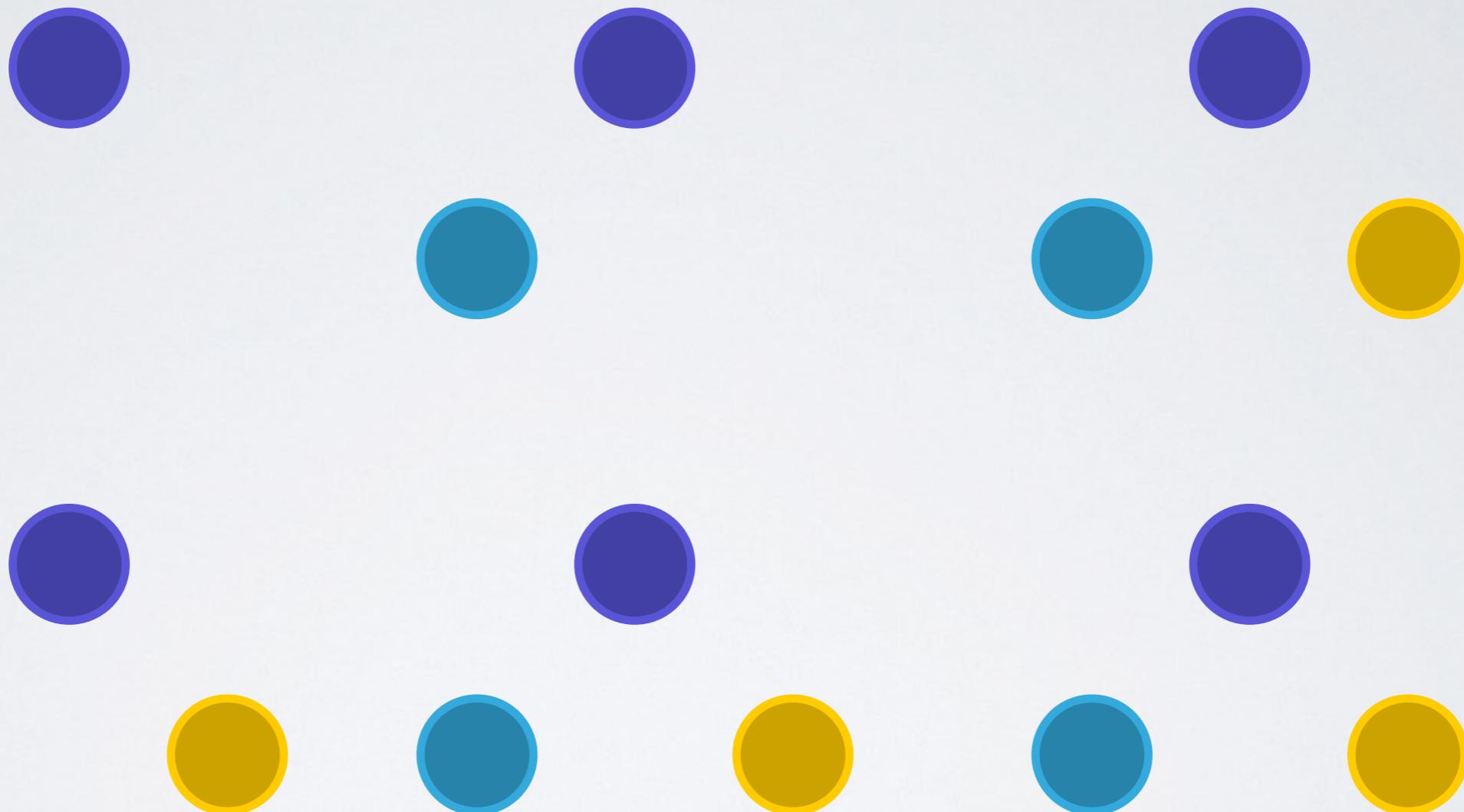
MULTIPLE MVCS



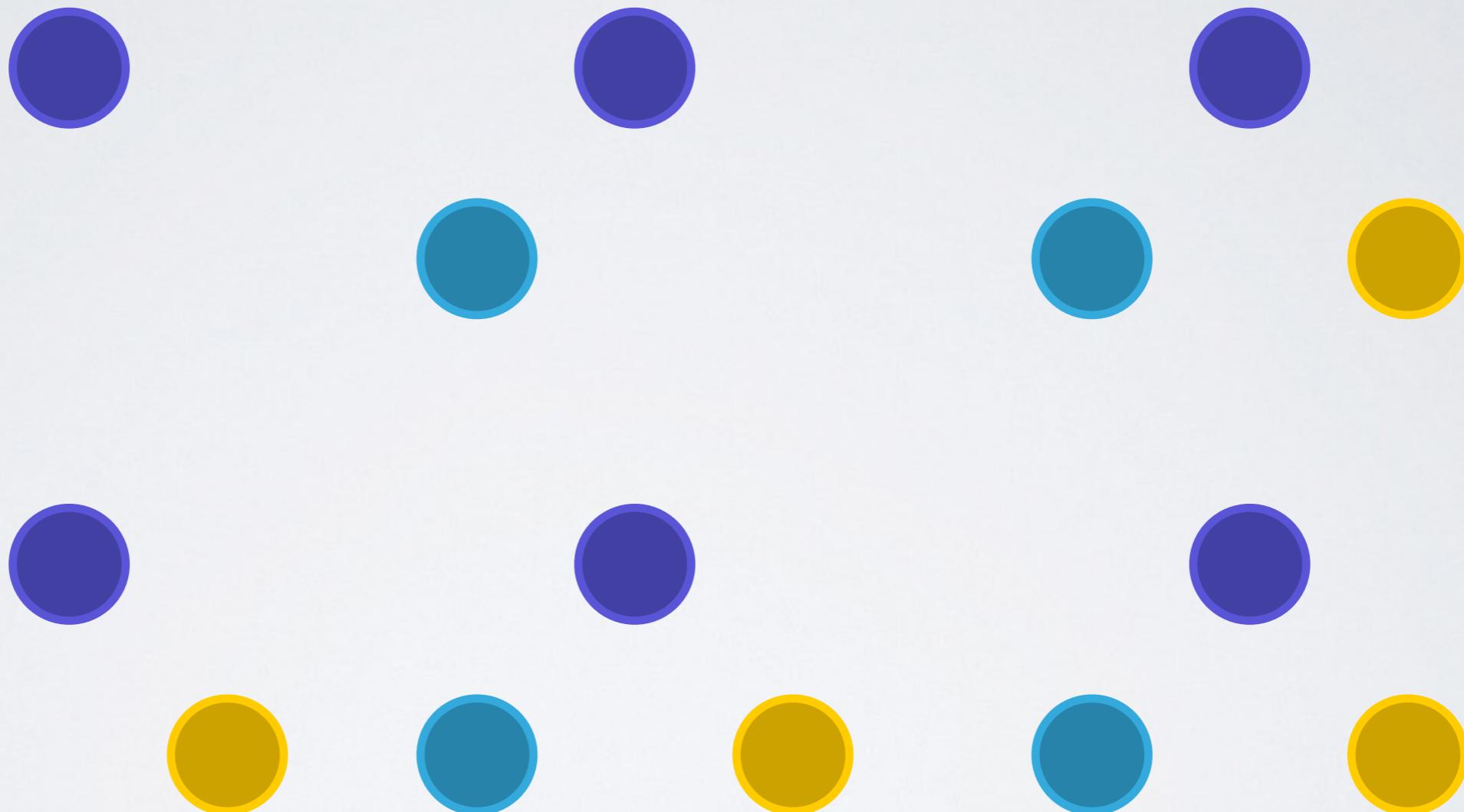
MULTIPLE MVCS



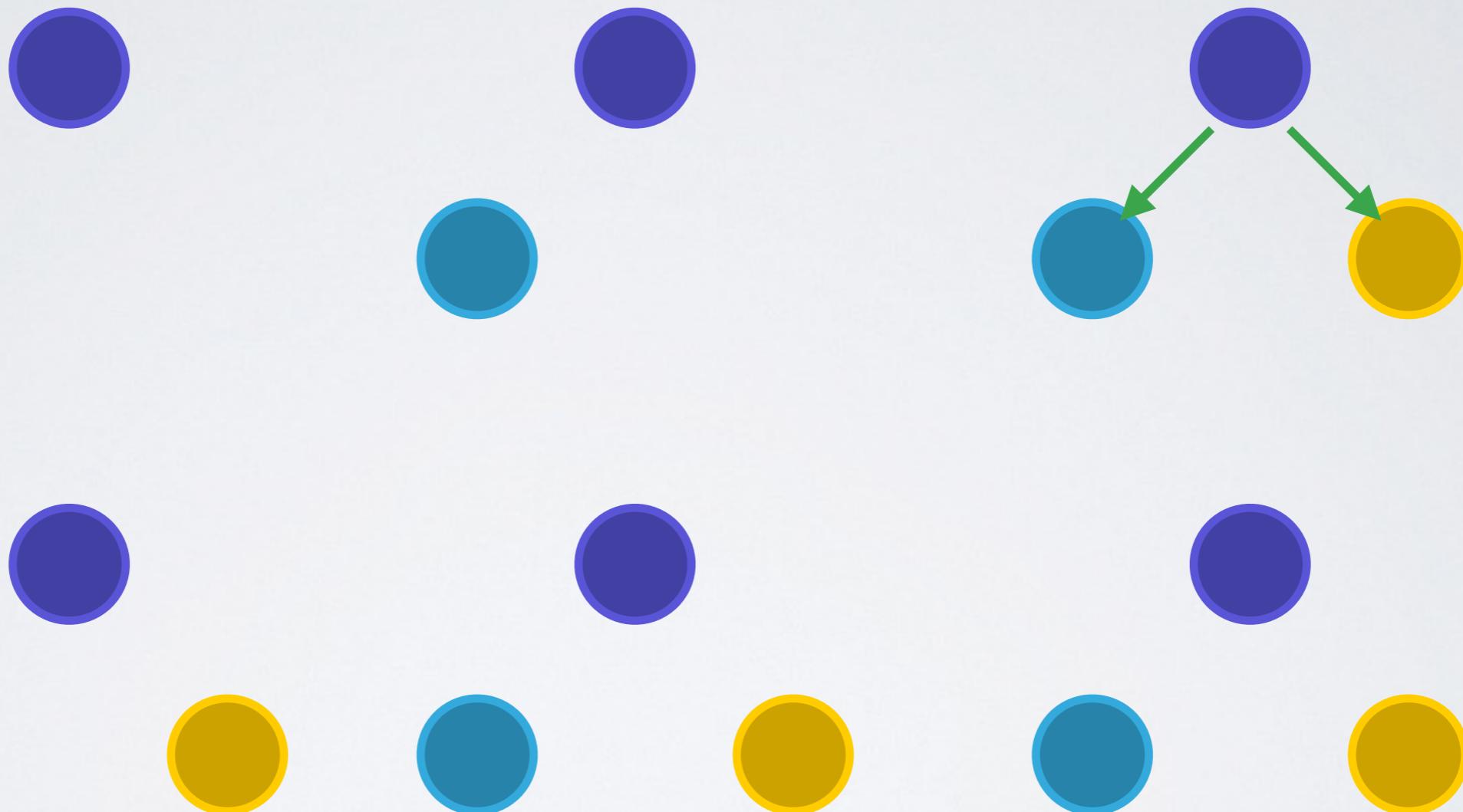
MULTIPLE MVCS



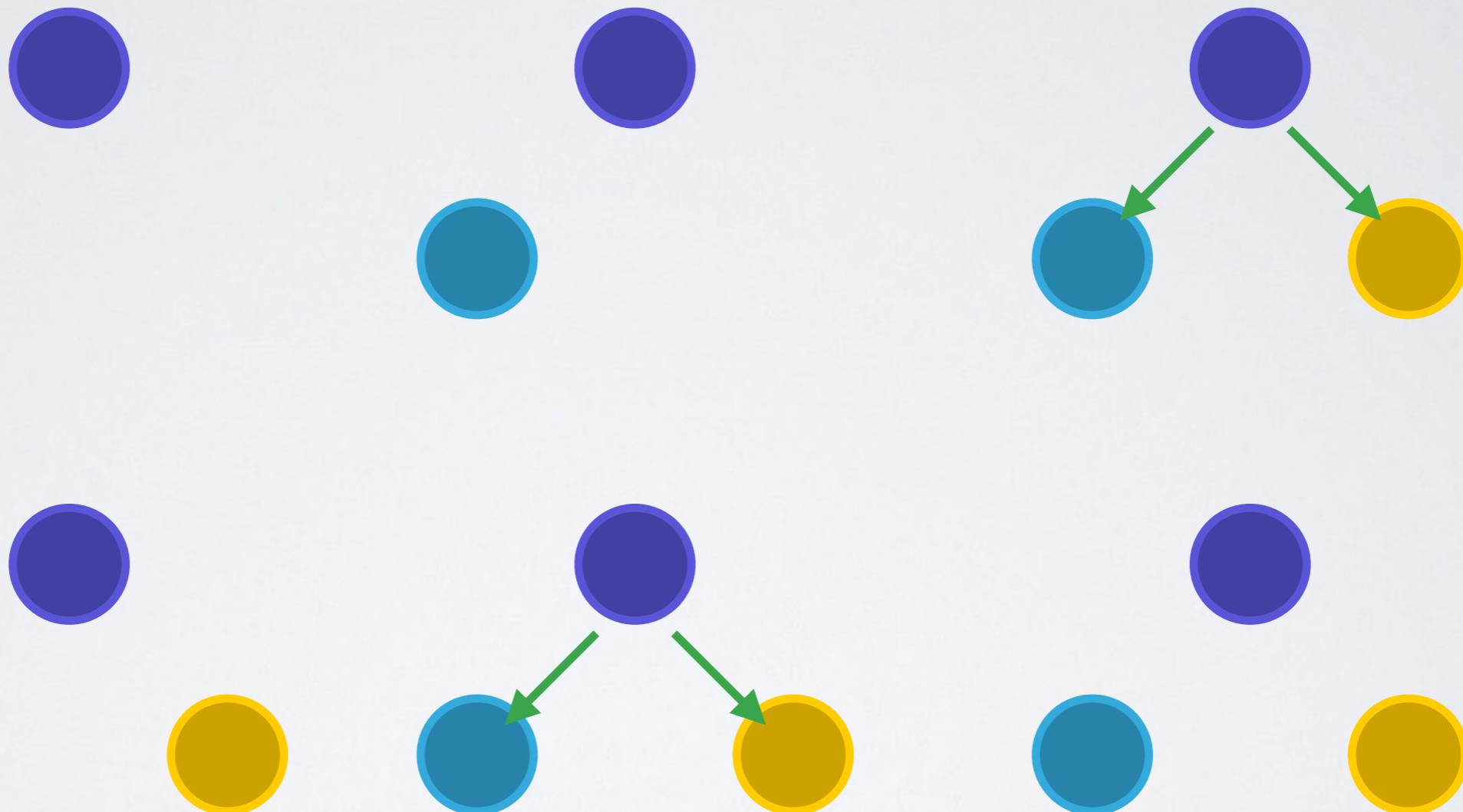
MULTIPLE MVCS



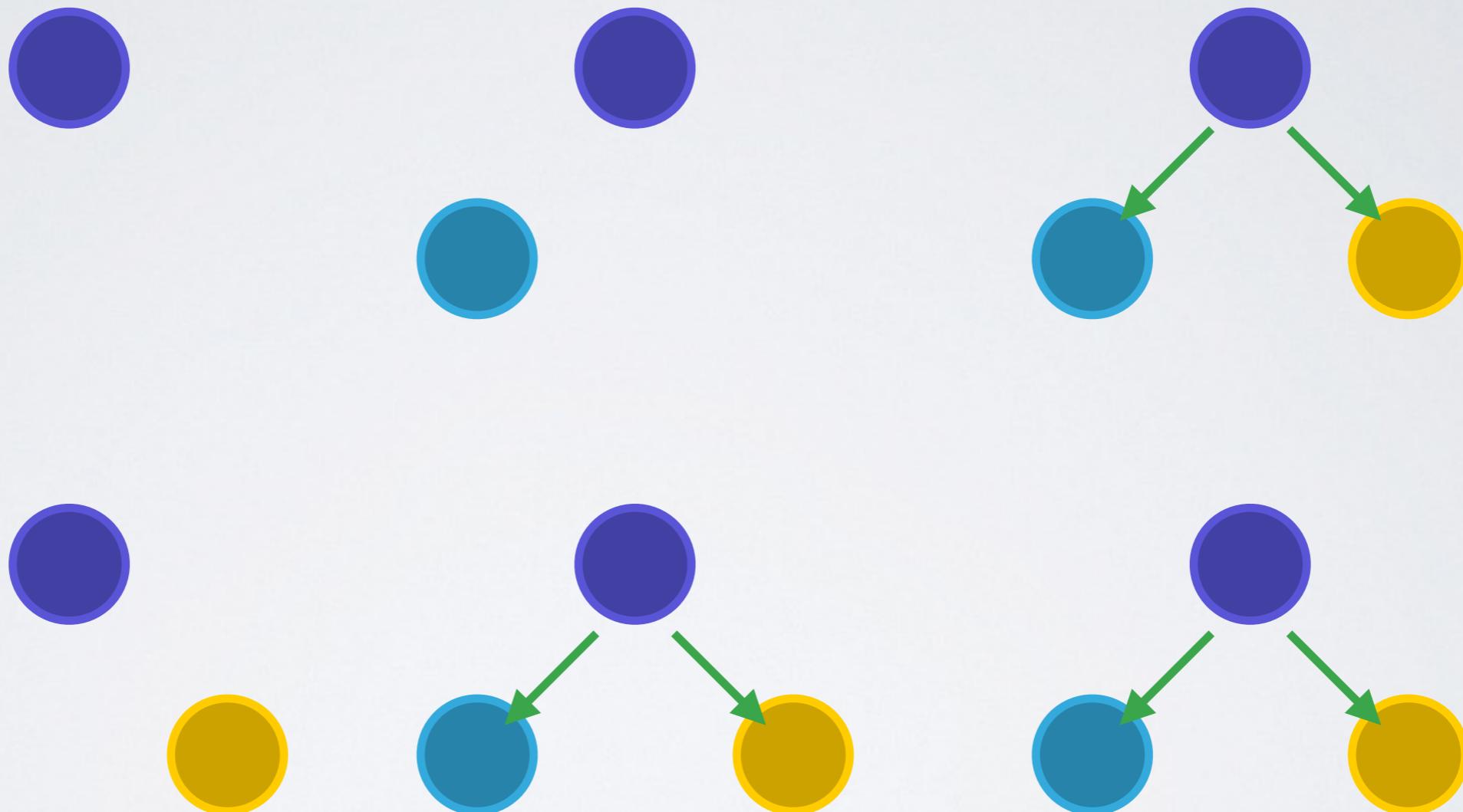
MULTIPLE MVCS



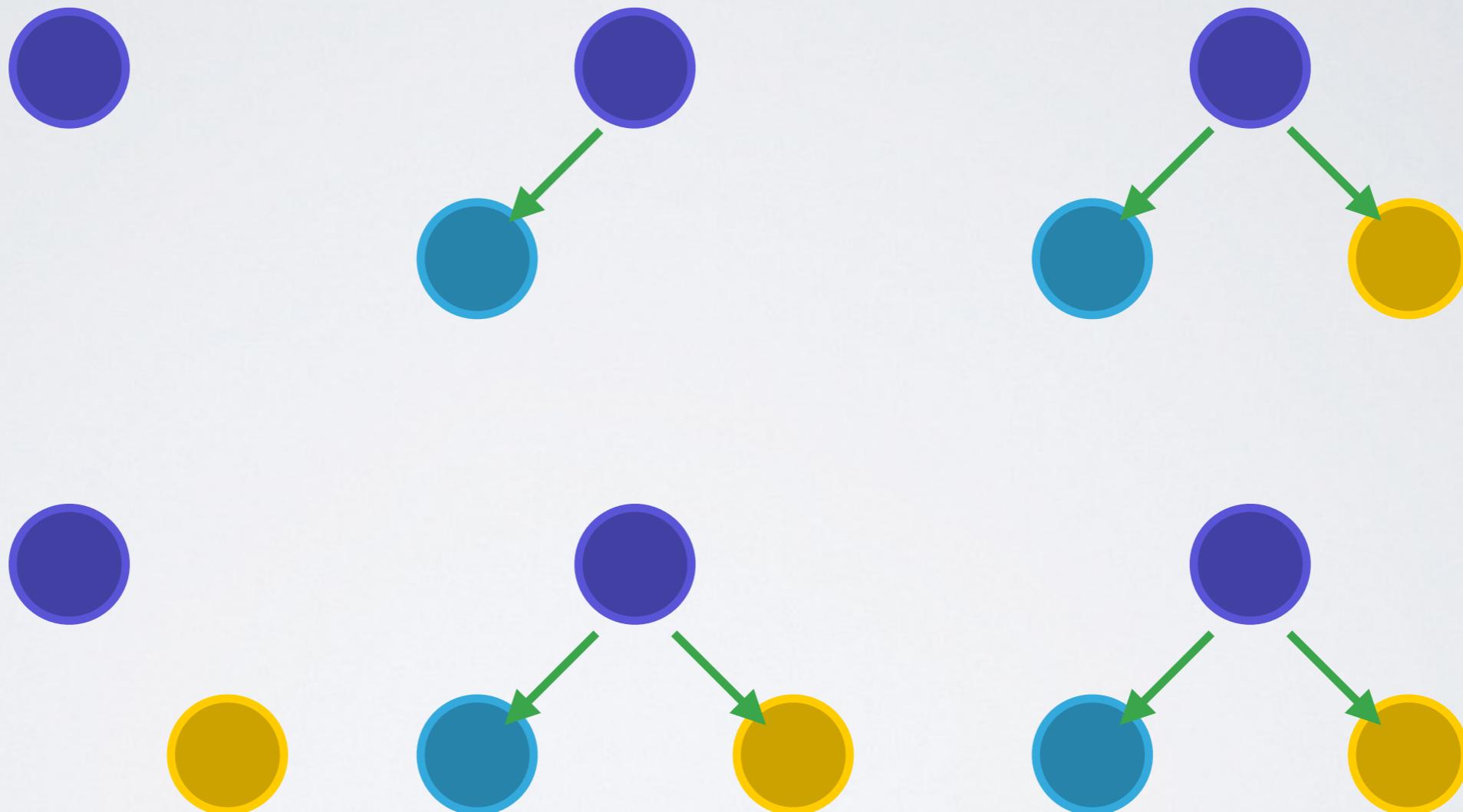
MULTIPLE MVCS



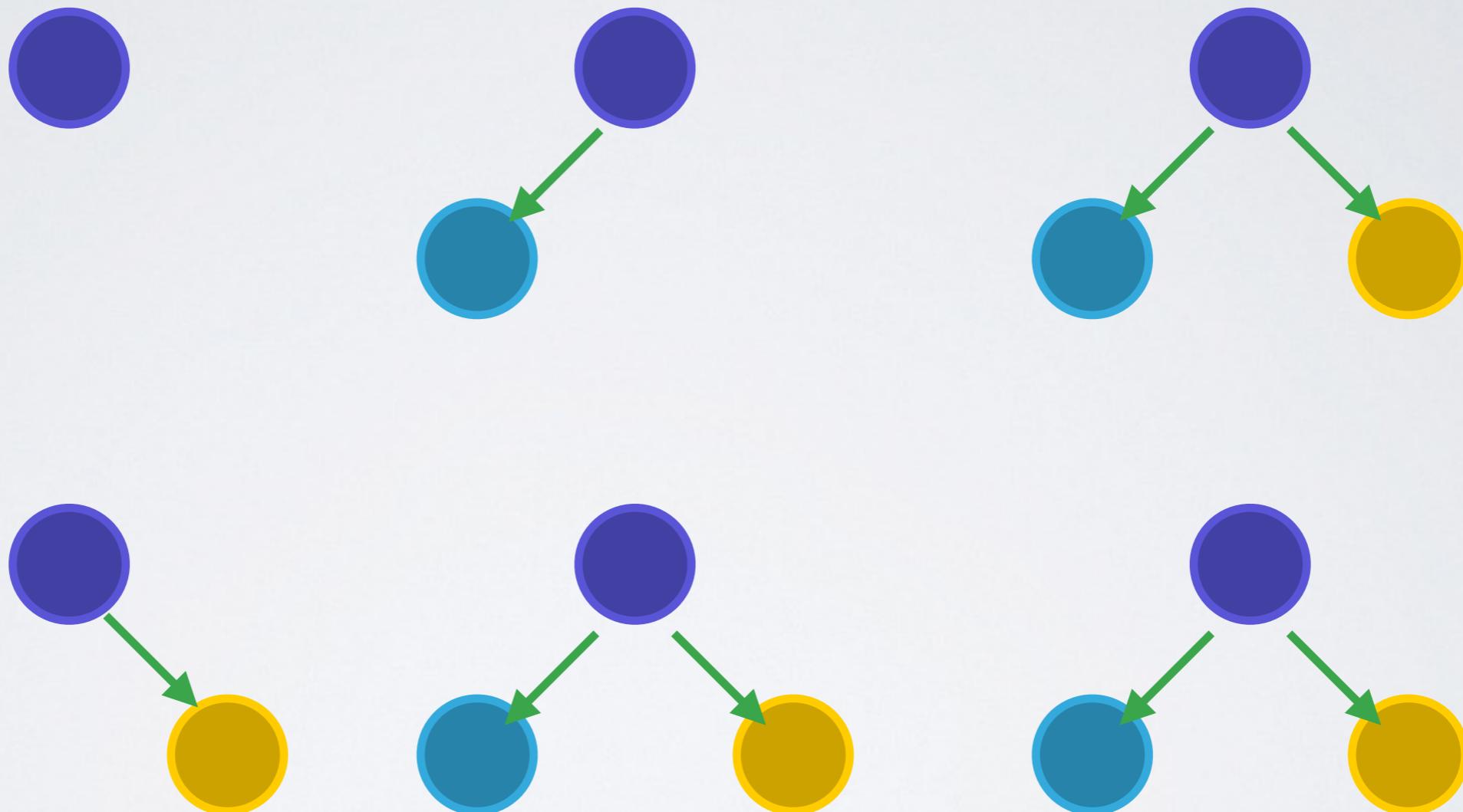
MULTIPLE MVCS



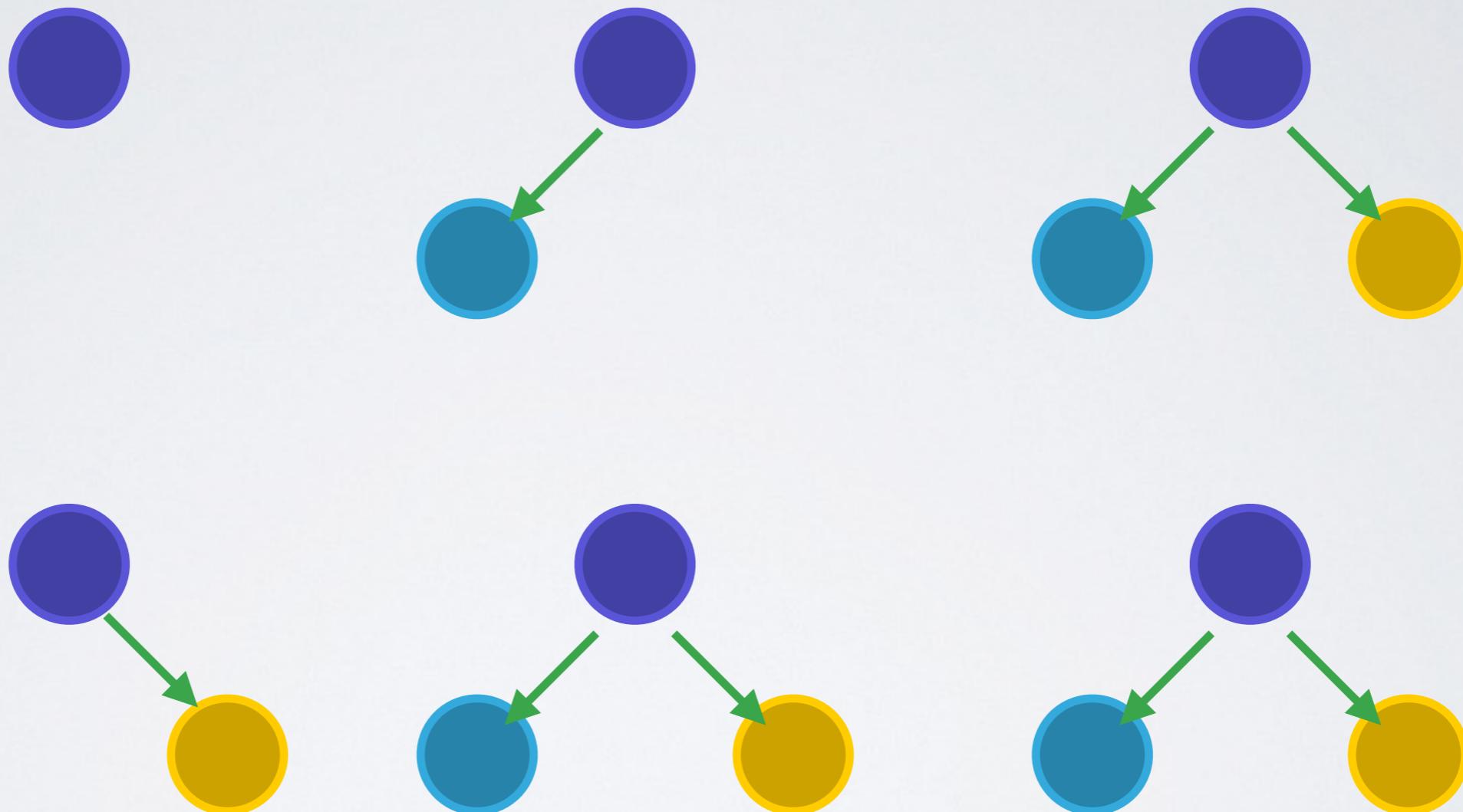
MULTIPLE MVCS



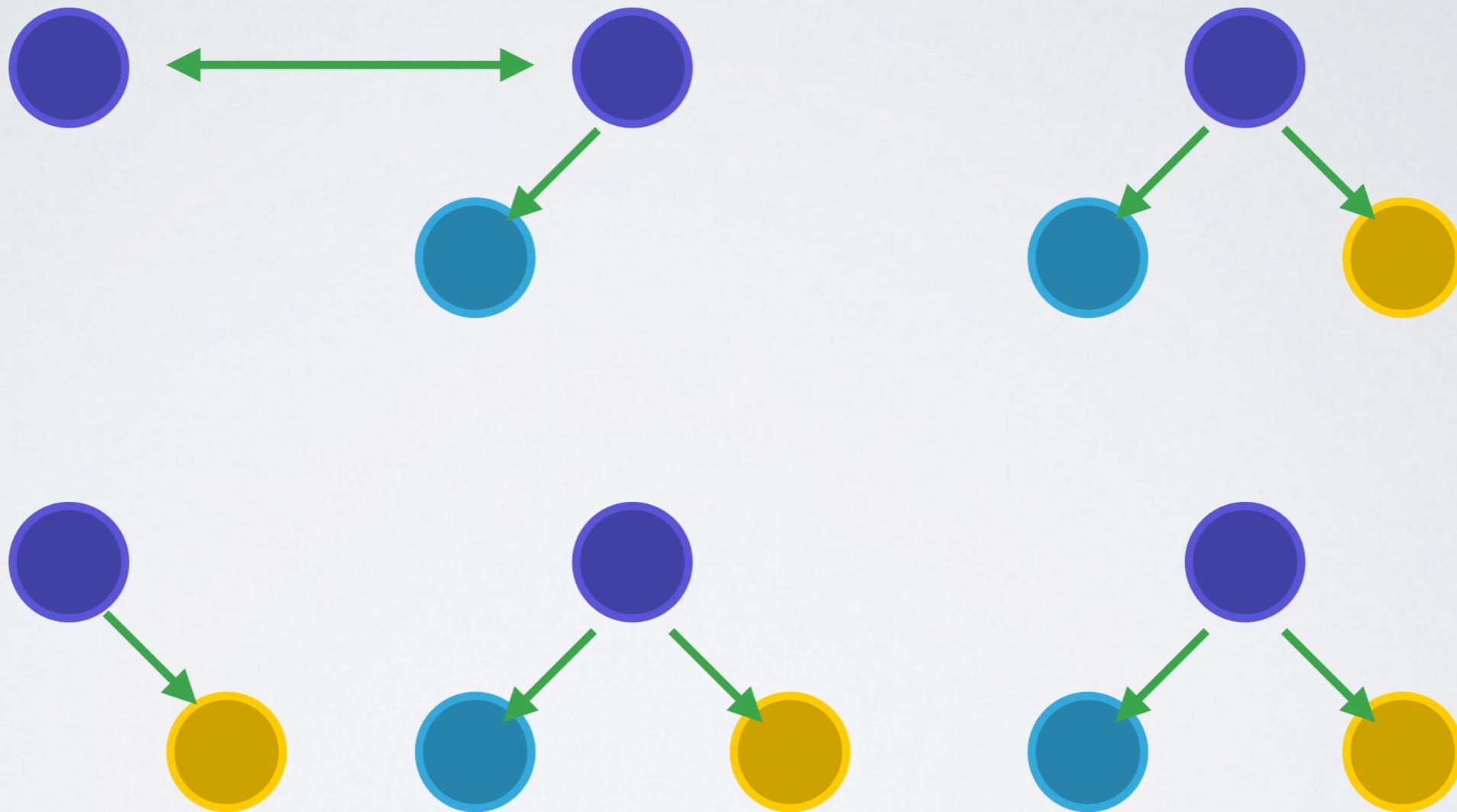
MULTIPLE MVCS



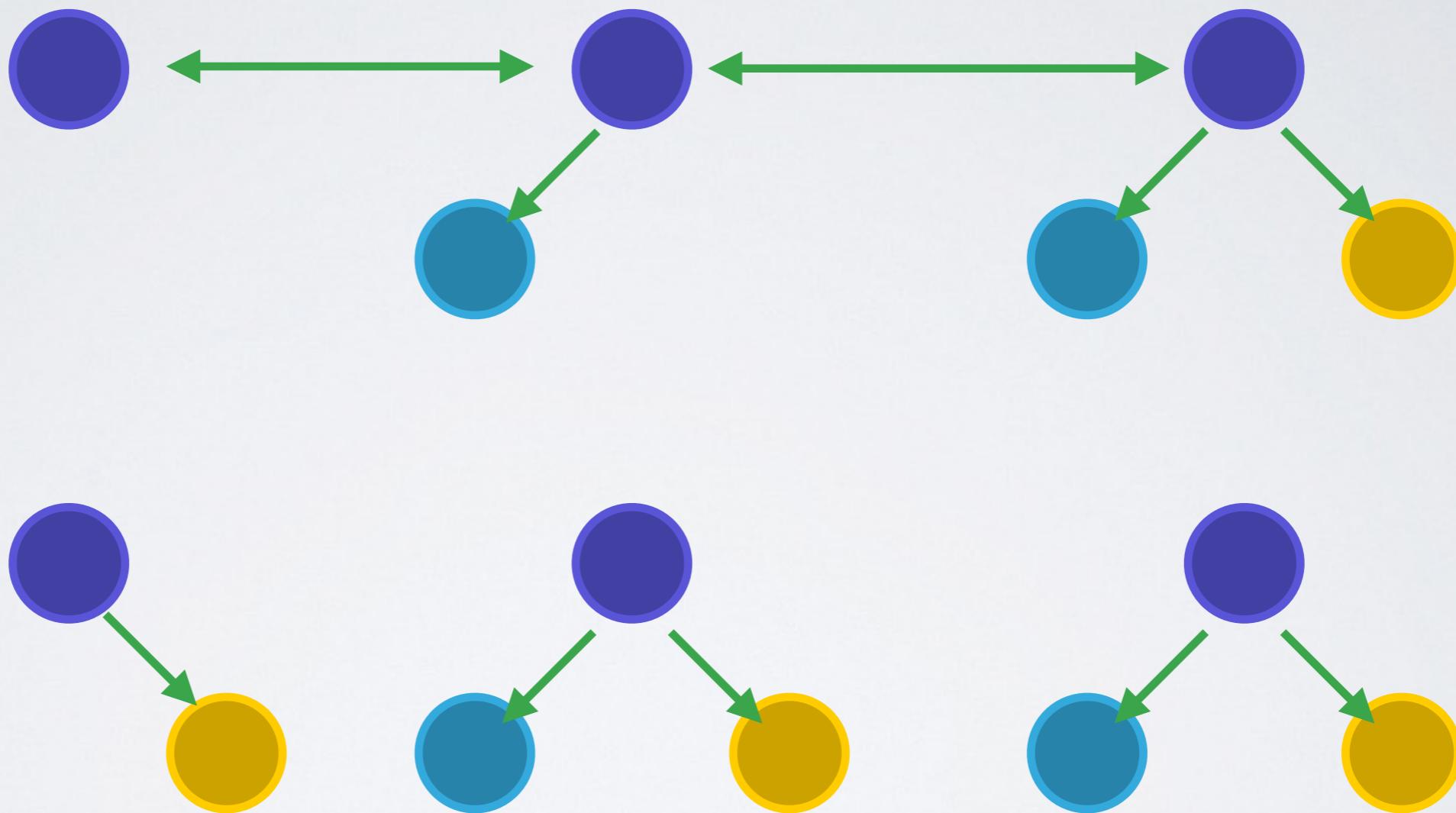
MULTIPLE MVCS



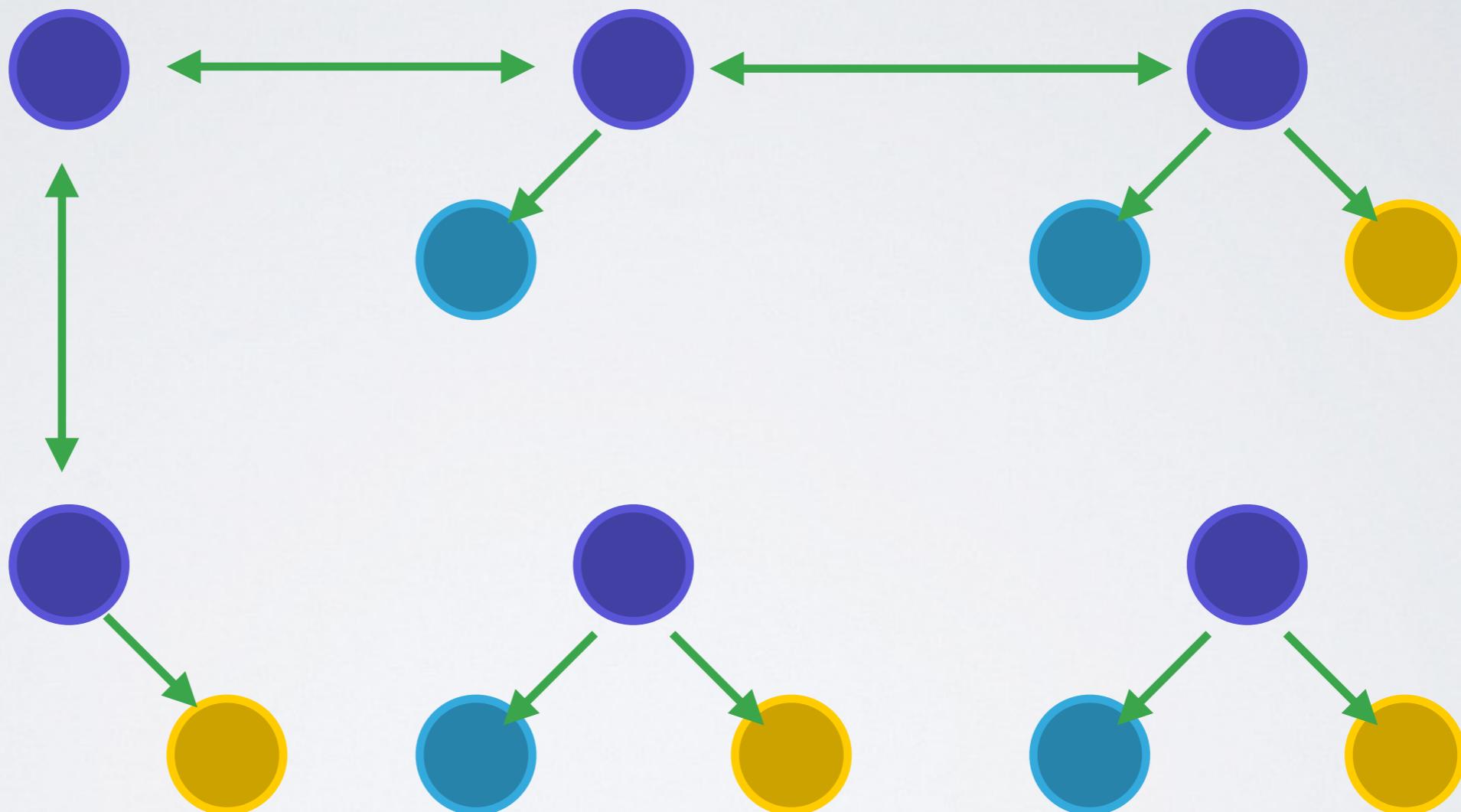
MULTIPLE MVCS



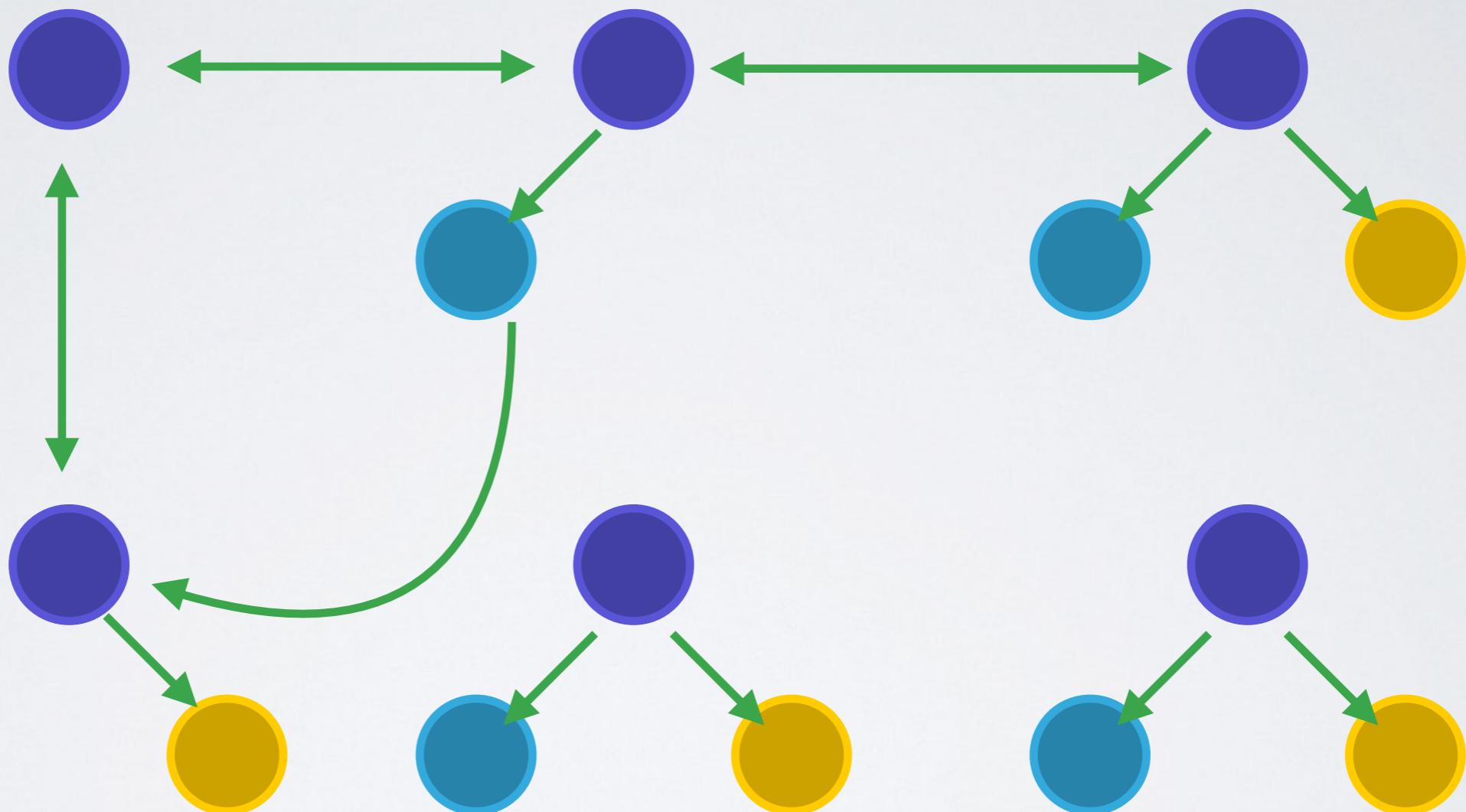
MULTIPLE MVCS



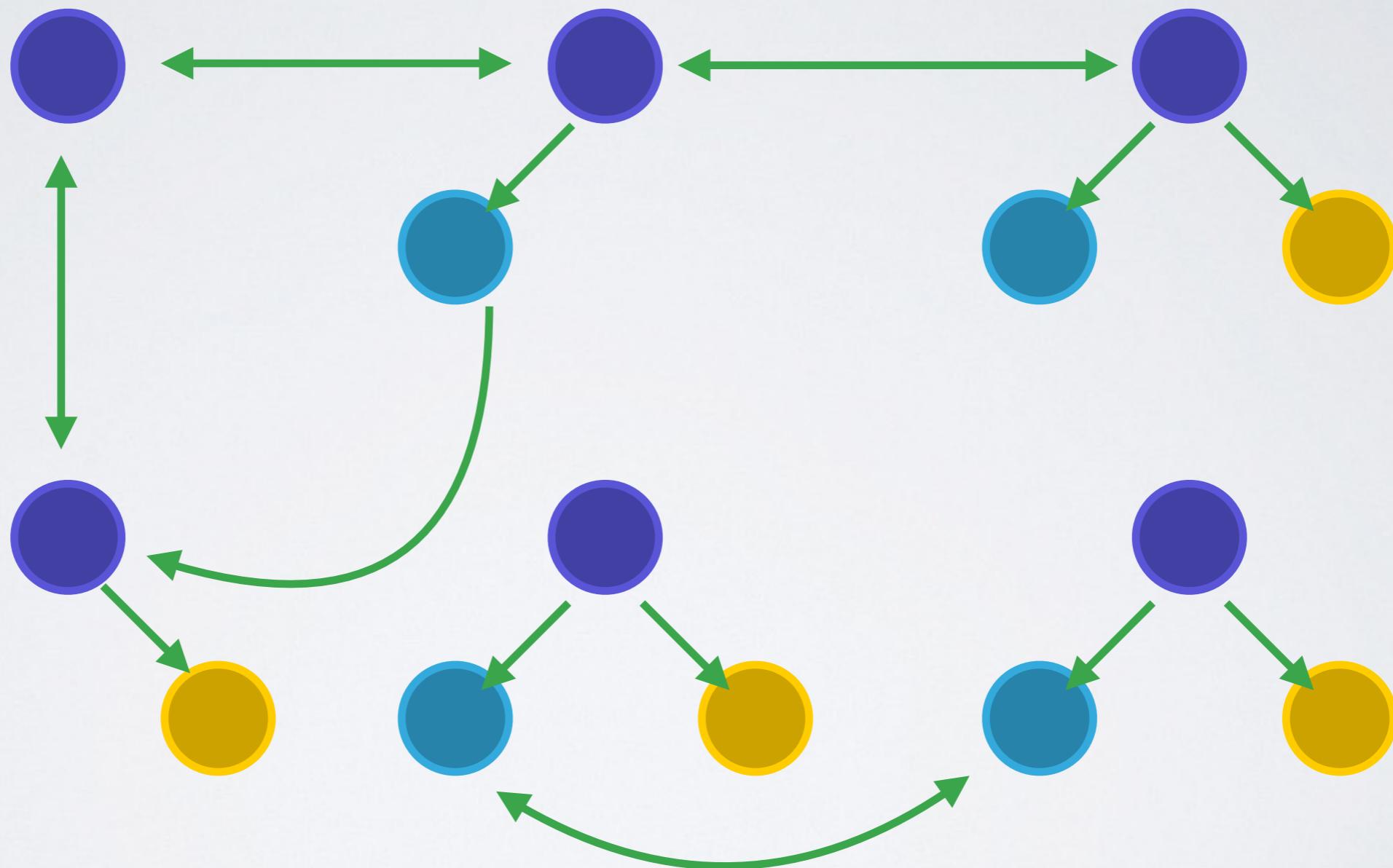
MULTIPLE MVCS



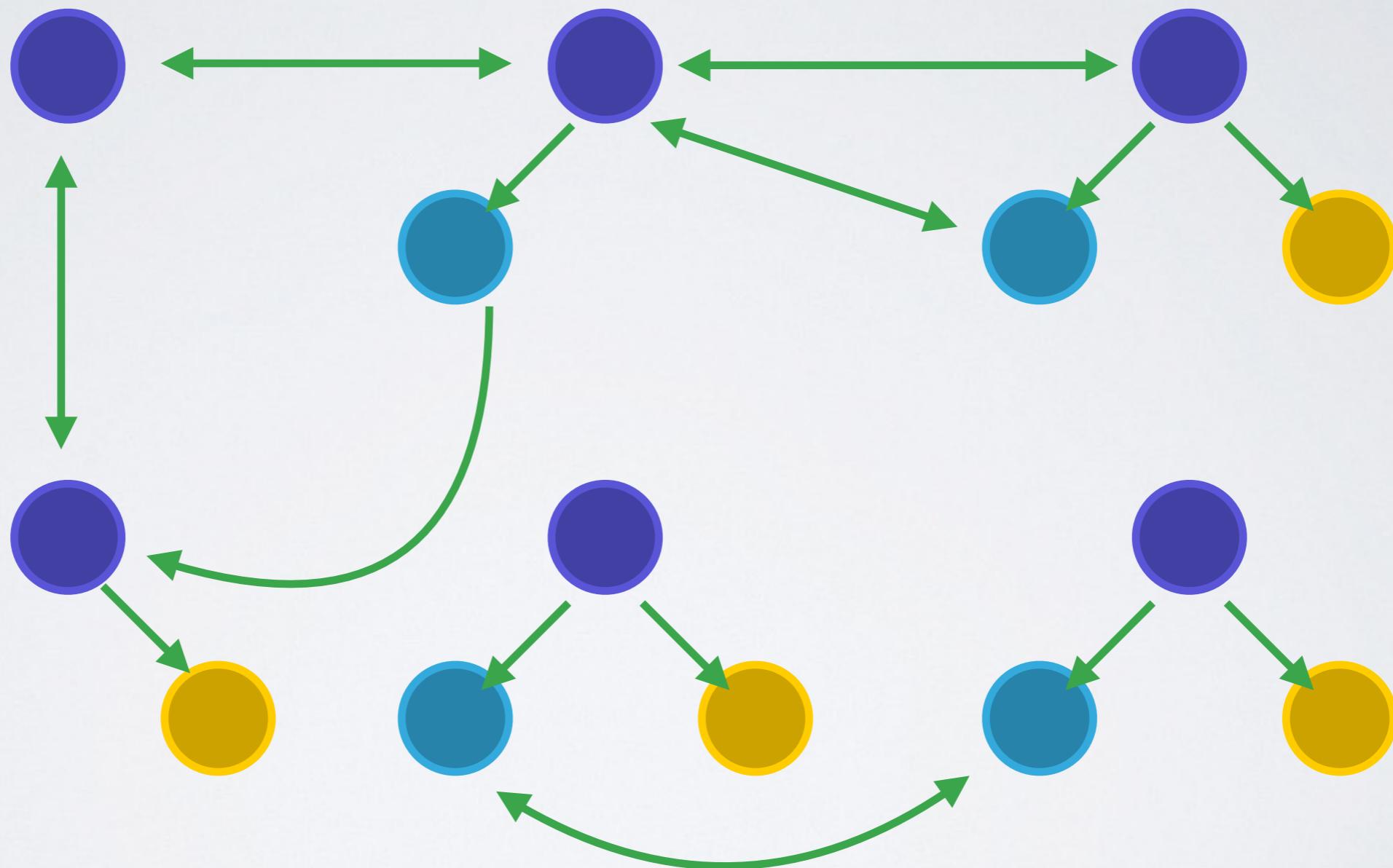
MULTIPLE MVCS



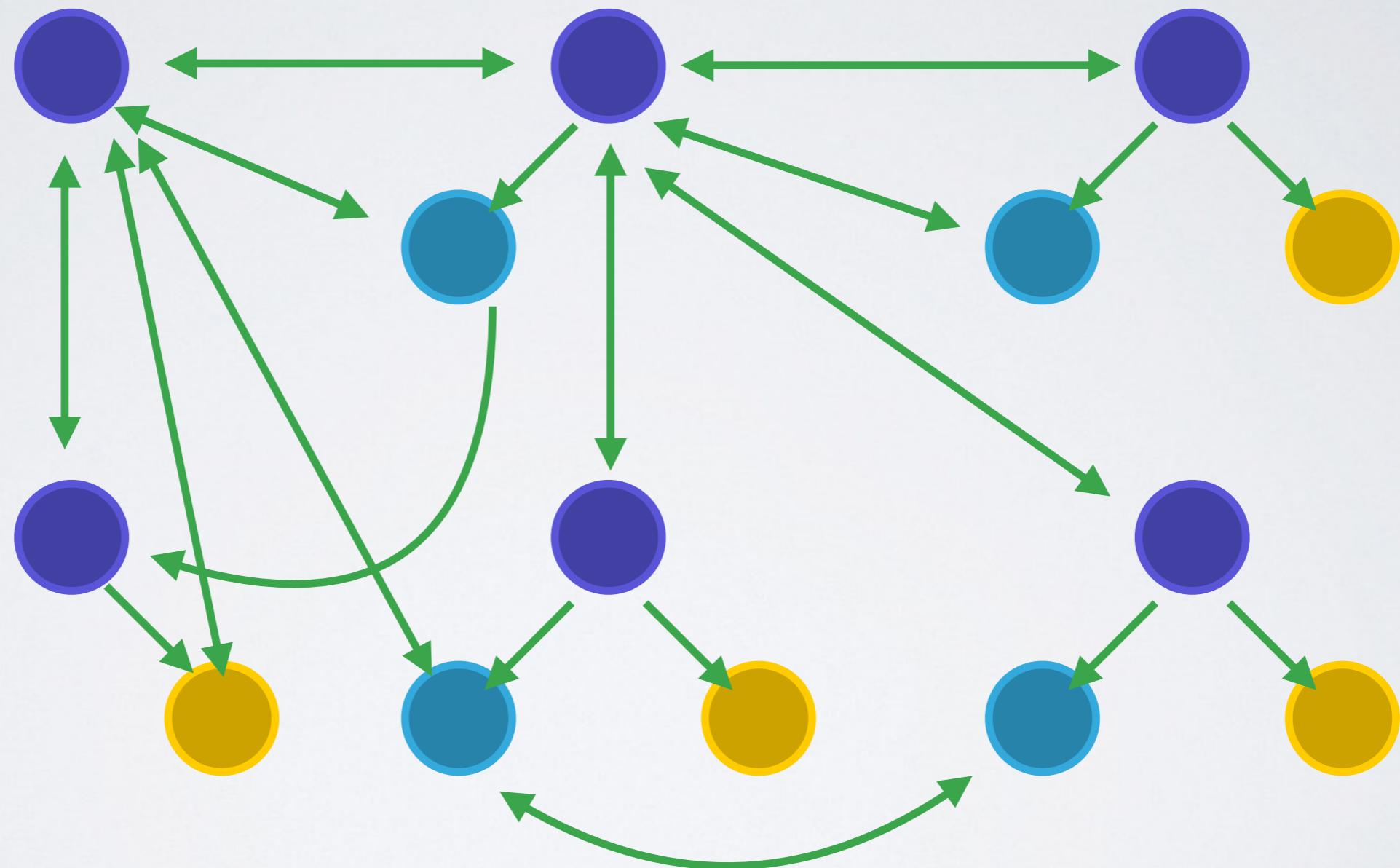
MULTIPLE MVCS



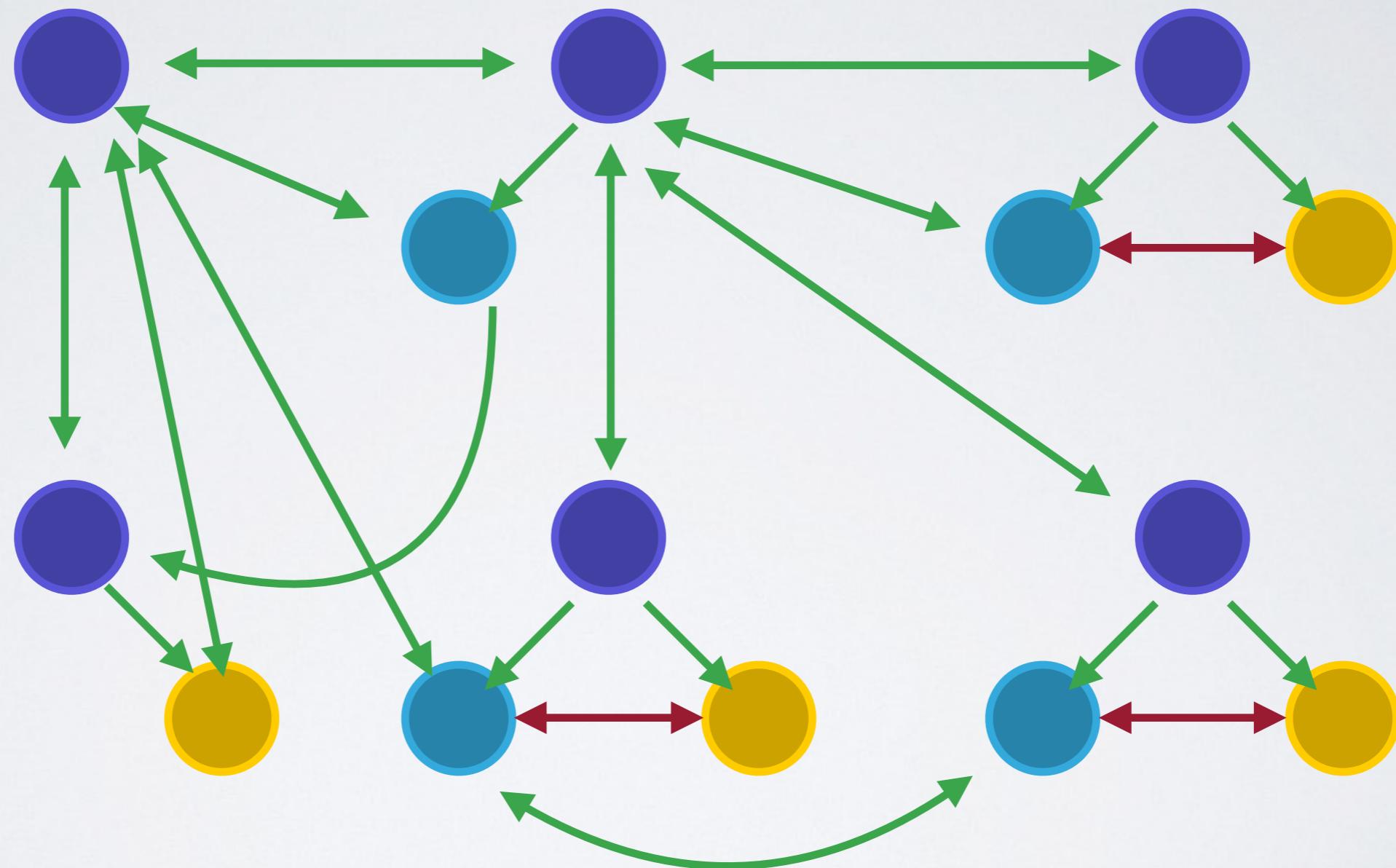
MULTIPLE MVCS



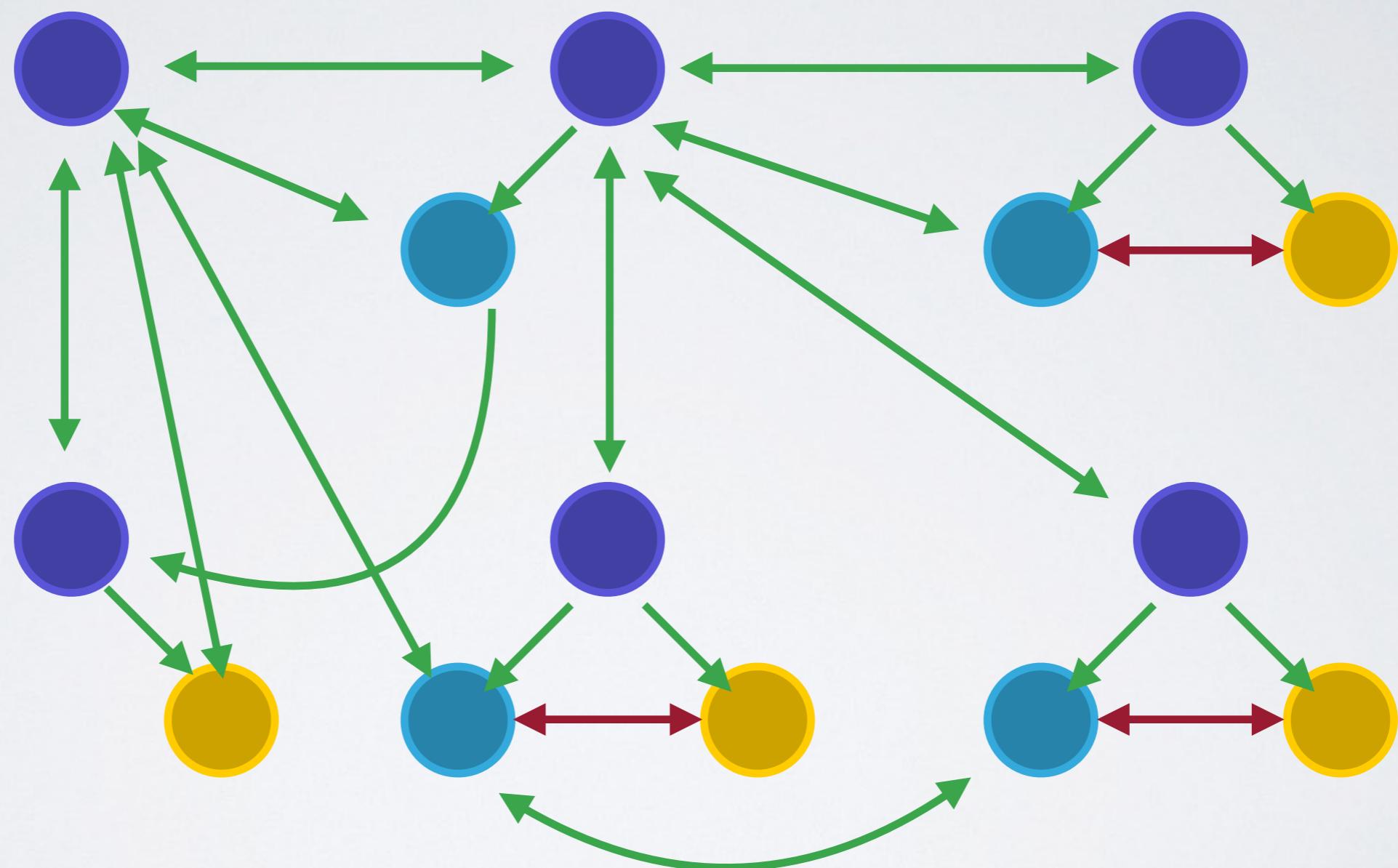
MULTIPLE MVCS



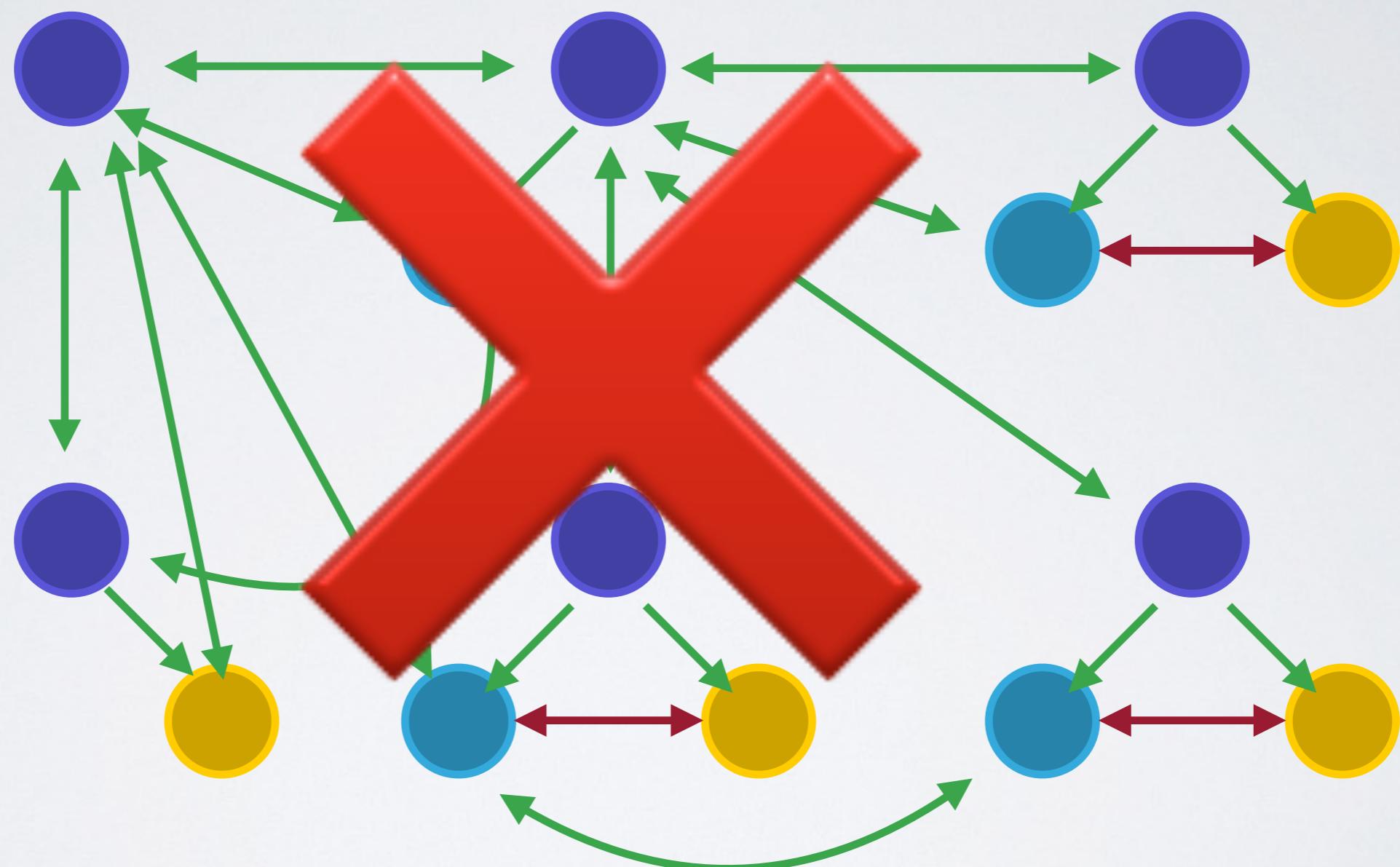
MULTIPLE MVCS



MULTIPLE MVCS FAILING



MULTIPLE MVCS FAILING



MVC (DETOUR)

martinfowler.com/eaaDev/uiArchs.html

GUI Architectures

There have been many different ways to organize the code for a rich client system. Here I discuss a selection of those that I feel have been the most influential and introduce how they relate to the patterns.

18 July 2006

 Martin Fowler

Contents

- Forms and Controls
- Model View Controller
- VisualWorks Application Model
- Model-View-Presenter (MVP)
- Humble View

This is part of the Further Enterprise Application Architecture development writing that I was doing in the mid 2000's. Sadly too

LET'S BUILD THIS USING MVC

Simulator - iPhone 6 - iPhone 6 / iOS 9.0 (13A4305g)



GitHub
GitHub, the company.
San Francisco, CA

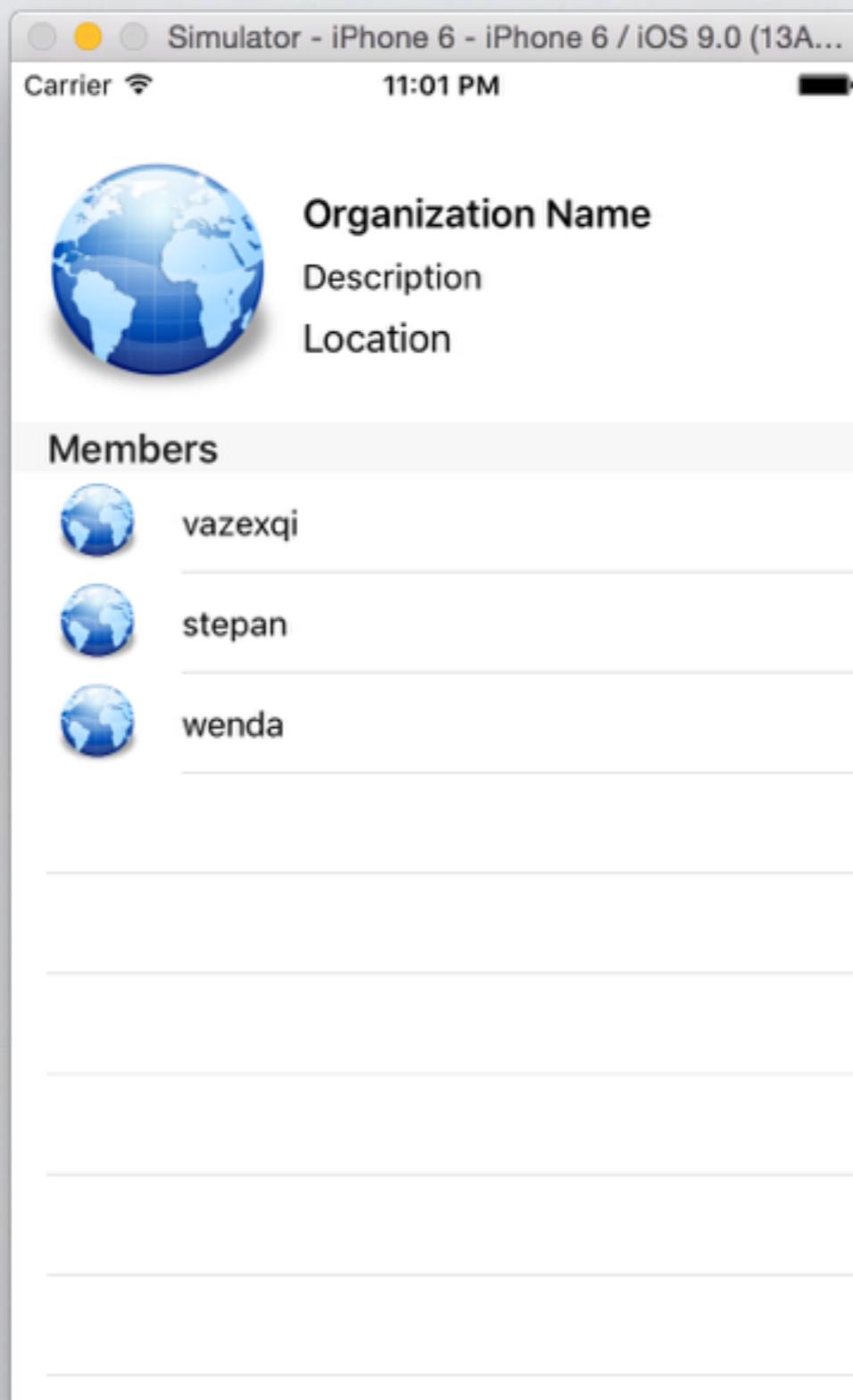
Members

Profile Picture	Member Name
	achiu
	adelcambre
	aden
	aitchabee
	alysonla

VIEW

- Let's prototype the design in Storyboards first.
- Get a better feel for table views and auto-layouts

TABLEVIEW (STATIC)



TABLEVIEW (STATIC)

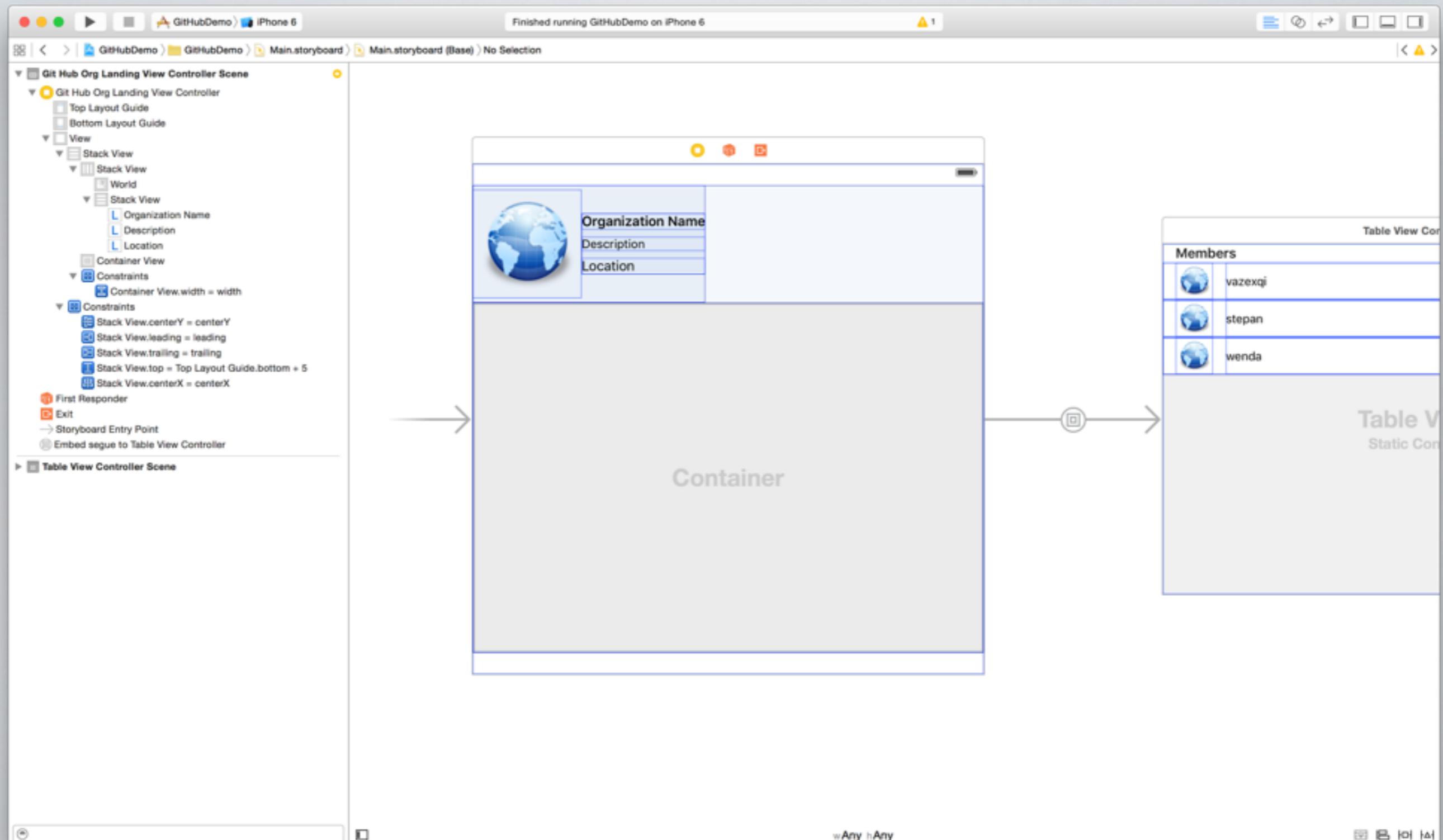
Simulator - iPhone 6 - iPhone 6 / iOS 9.0 (13A4305g)

Organization Name
Description
Location

Members

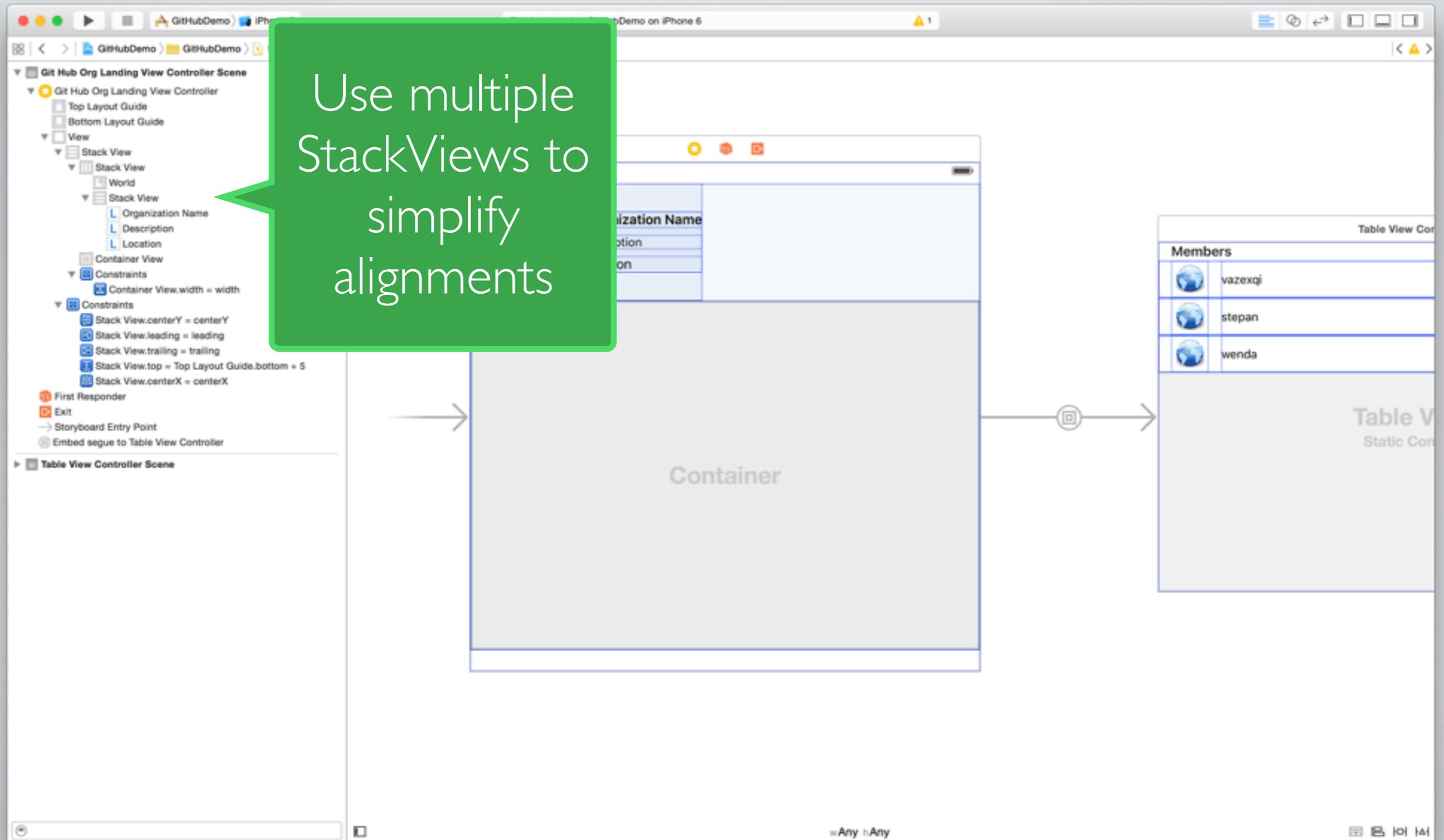
vazexqi
stepan
wenda

TABLEVIEW (STATIC)

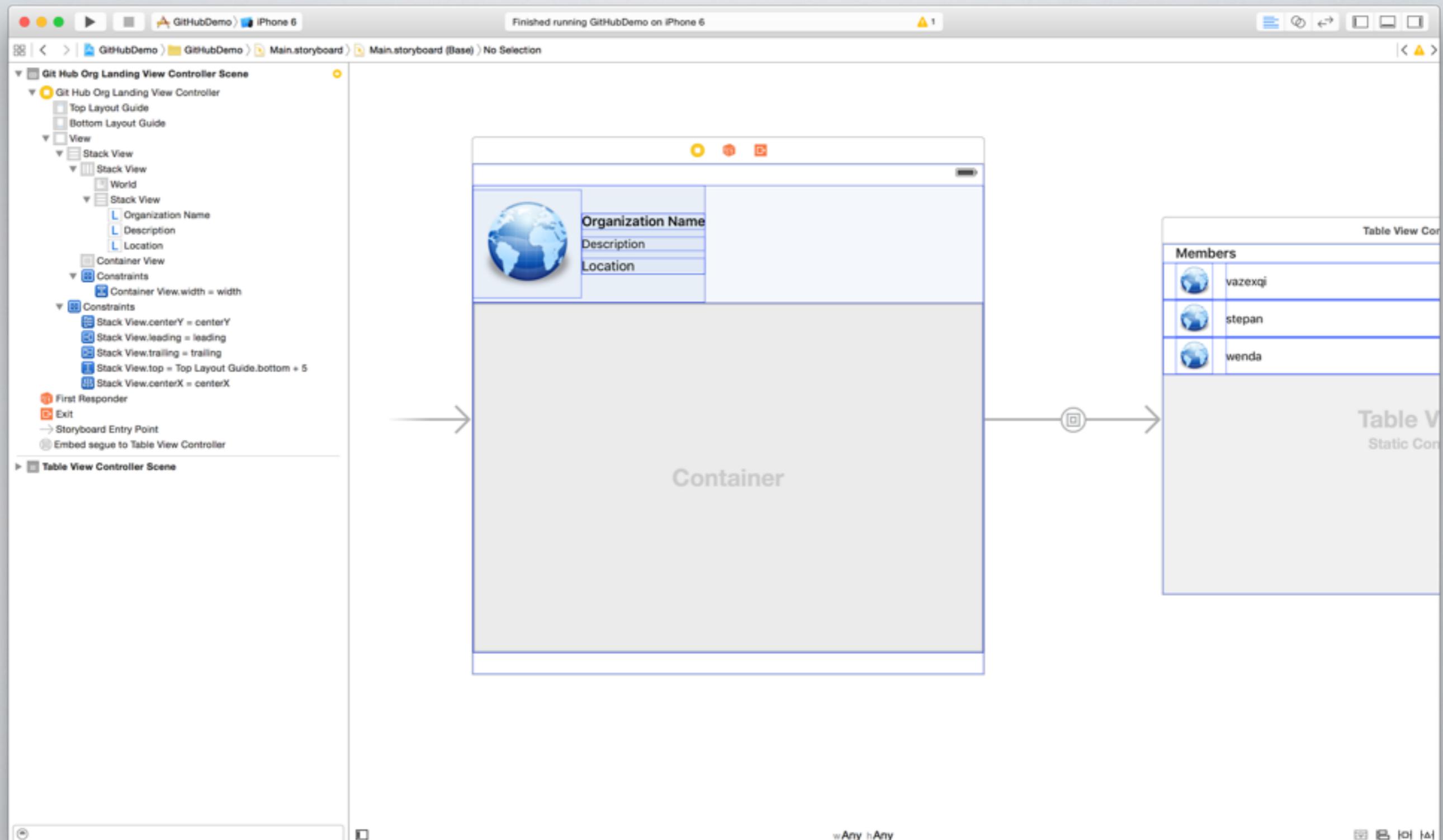


TABLEVIEW (STATIC)

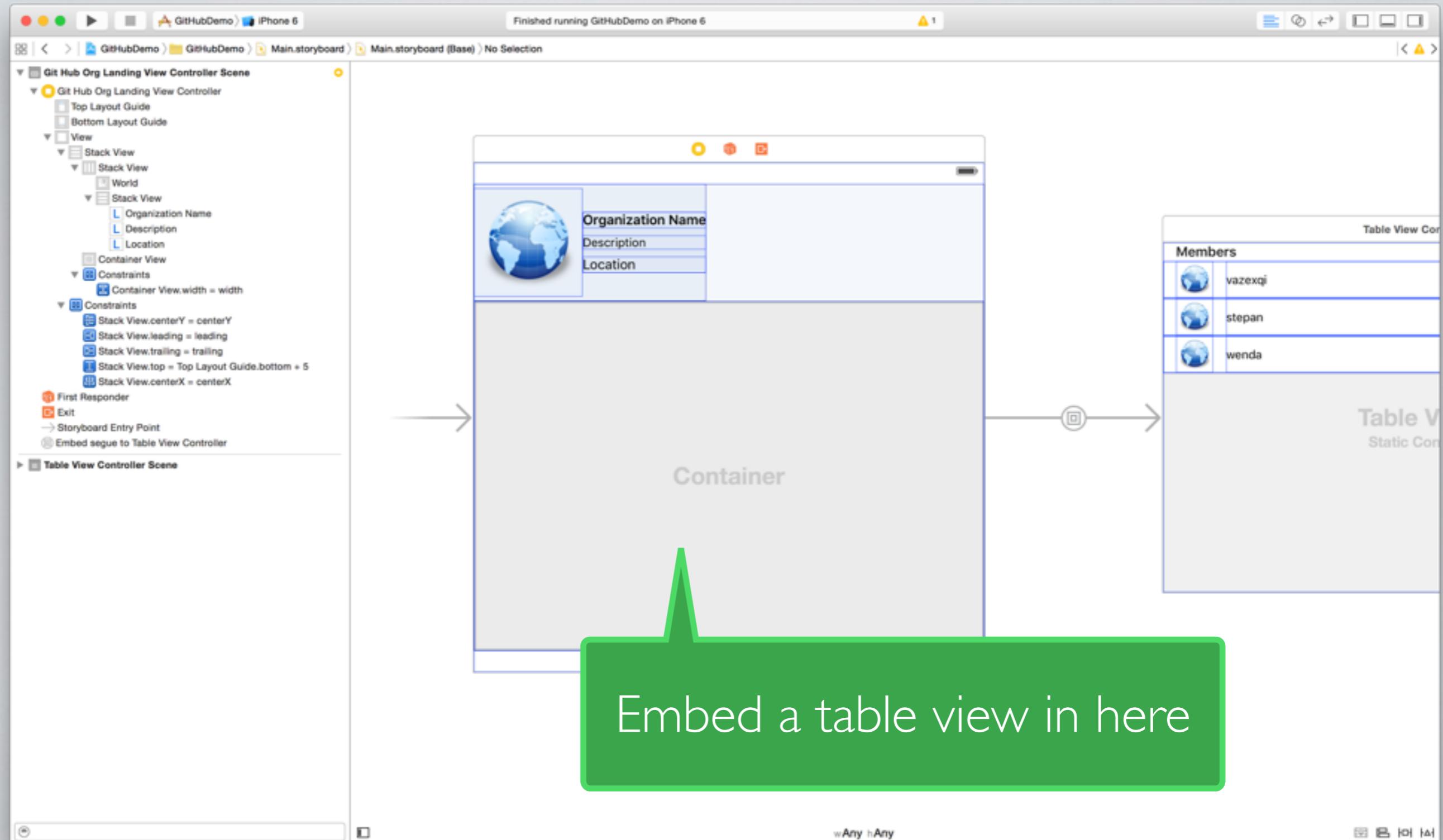
Use multiple
StackViews to
simplify
alignments



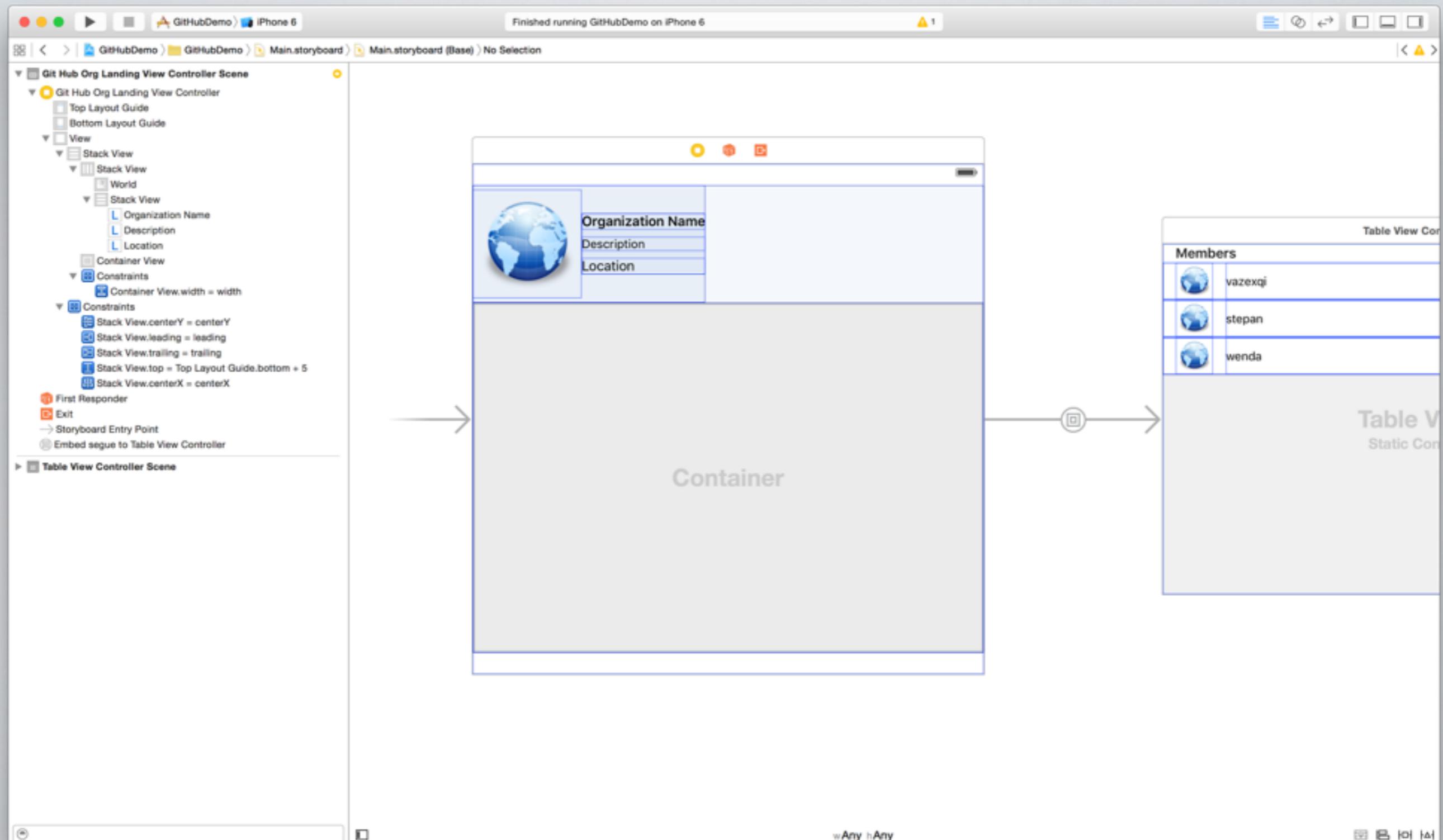
TABLEVIEW (STATIC)



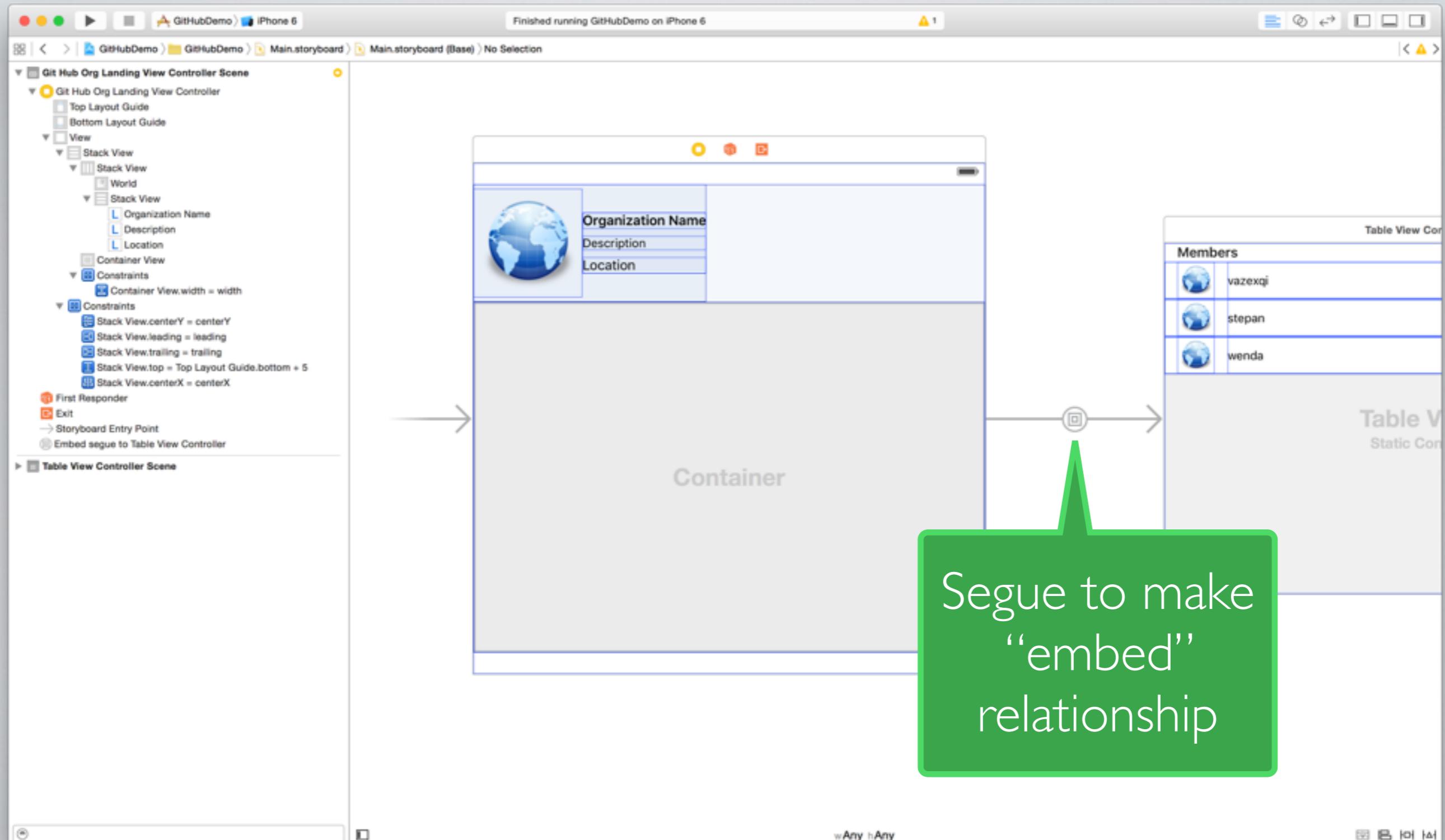
TABLEVIEW (STATIC)



TABLEVIEW (STATIC)



TABLEVIEW (STATIC)



HANDS-ON

MODEL

We need some way to get the data from GitHub

GITHUB API

The screenshot shows a web browser window with the URL `developer.github.com/v3/orgs/#get-an-organization`. The page title is "Get an organization". Below the title, there is a code block for a GET request:

```
GET /orgs/:org
```

Under the "Response" section, the status code and rate limit information are shown:

```
Status: 200 OK
X-RateLimit-Limit: 5000
X-RateLimit-Remaining: 4999
```

Below that, the JSON response body is displayed:

```
{
  "login": "github",
  "id": 1,
  "url": "https://api.github.com/orgs/github",
  "avatar_url": "https://github.com/images/error/octocat_happy.gif",
  "description": "A great organization",
  "name": "github",
  "company": "GitHub",
  "blog": "https://github.com/blog",
  "location": "San Francisco",
  "email": "octocat@github.com",
```

GITHUB API

Simulator - iPhone 6 - iPhone 6 / iOS 9.0 (13A4305g)



GitHub
GitHub, the company.
San Francisco, CA

Members

	achiu
	adelcambre
	aden
	aitchabee
	alysonla

GITHUB API

Simulator - iPhone 6 - iPhone 6 / iOS 9.0 (13A4305g)



GitHub
GitHub, the company.
San Francisco, CA

Organization

Members

Profile Picture	Username
	achiu
	adelcambre
	aden
	aitchabee
	alysonla

GITHUB API

Simulator - iPhone 6 - iPhone 6 / iOS 9.0 (13A4305g)

The image shows an iPhone 6 simulator running iOS 9.0. The screen displays the GitHub organization profile for "GitHub". The profile includes the GitHub logo, the name "GitHub", the company name "GitHub, the company.", and the location "San Francisco, CA". A green callout bubble with a white arrow points to the word "Organization". Below the profile, a section titled "Members" lists five GitHub users with their profile pictures and names: "achiu", "adelcambre", "aden", "aitchabee", and "alysonla". A green callout bubble with a white arrow points to the word "Members".

GitHub
GitHub, the company.
San Francisco, CA

Organization

Members

	achiu
	adelcambre
	aden
	aitchabee
	alysonla

Members

GITHUB API

Screenshot of the Paw API client showing a successful GET request to the GitHub API.

Request URL: :GET https://api.github.com/orgs/github

Status: 200 OK

Headers:

Header Name	Header Value
Add Header Name	Add Header Value

Response:

Key or Index	Type	Value
Root	22 items	
login	A	github
id	42	9919
url	A	https://api.github.com/orgs/github
repos_url	A	https://api.github.com/orgs/github/repos
events_url	A	https://api.github.com/orgs/github/events
members_url	A	https://api.github.com/orgs/github/m...
public_members_url	A	https://api.github.com/orgs/github/pu...
avatar_url	A	https://avatars.githubusercontent.com...
description	A	GitHub, the company.
name	A	GitHub
company	null	
blog	A	https://github.com/about
location	A	San Francisco, CA
email	A	support@github.com
public_repos	42	117
public_gists	42	0

Tools: Swift (Alamofire) ▾ Copy Export Open With ▾



GITHUB API

Paw GitHubSession Paw

Cookies Default Jar
Environments 1 »
Organization GET https://api.github.com/orgs/github 200 OK
Members GET https://api.github.com/orgs/github... 200 OK

Organization

: GET https://api.github.com/orgs/github

Info Request Response Headers JSON Raw

Key or Index Type Value

Root 22 items

login	A github
id	42 9919
url	A https://api.github.com/orgs/github
repos_url	A https://api.github.com/orgs/github/repos
events_url	A https://api.github.com/orgs/github/events
members_url	A https://api.github.com/orgs/github/m...
public_members_url	A https://api.github.com/orgs/github/pu...
avatar_url	A https://avatars.githubusercontent.com...
description	A GitHub, the company.
name	A GitHub
company	null
blog	A https://github.com/about
location	A San Francisco, CA
email	A support@github.com
public_repos	42 117
public_gists	42 0

Swift (Alamofire) Copy Export Open With



GITHUB API

Screenshot of Paw (API testing tool) showing a successful GET request to the GitHub API.

Request URL: :GET https://api.github.com/orgs/github/members

Status: 200 OK

Options:

- Timeout: No timeout
- Redirections:
 - Follow Redirects
 - Redirect Authorization
 - Redirect Original Method
- Cookies:
 - Automatically Send Cookies
 - Store Received Cookies
- Authorization:
 - Set HTTP Basic Auth
 - Set OAuth 1
 - Set OAuth 2
 - Set Amazon S3 Signature

Response Headers:

Key or Index	Type	Value
INDEX 0	17 items	
login	A	achiu
id	A	24772
avatar_url	A	https://avatars.githubusercontent.com/u/24772
gravatar_id	A	
url	A	https://api.github.com/users/achiu
html_url	A	https://github.com/achiu
followers_url	A	https://api.github.com/users/achiu/followers
following_url	A	https://api.github.com/users/achiu/following
gists_url	A	https://api.github.com/users/achiu/gists
starred_url	A	https://api.github.com/users/achiu/starred
subscriptions_url	A	https://api.github.com/users/achiu/subscriptions
organizations_url	A	https://api.github.com/users/achiu/orgs
repos_url	A	https://api.github.com/users/achiu/repos
events_url	A	https://api.github.com/users/achiu/events
received_events_url	A	https://api.github.com/users/achiu/received_events
type	A	User
site_admin	B1	true

Tool Buttons:

- Swift (Alamofire)
- Copy
- Export
- Open With



GITHUB API

Paw GitHubSession Paw

Cookies Default Jar
Environments 1 »

Organization
GET https://api.github.com/orgs/github
200 OK

Members
GET https://api.github.com/orgs/github/members
200 OK

: GET https://api.github.com/orgs/github/members

Headers URL Params Body Options

Info Request Response Headers JSON Raw

200 OK

Members

Key or Index	Type	Value
0	17 ITEMS	
login	A	achiu
id	A	24772
avatar_url	A	https://avatars.githubusercontent.com/u/24772
gravatar_id	A	
url	A	https://api.github.com/users/achiu
html_url	A	https://github.com/achiu
followers_url	A	https://api.github.com/users/achiu/followers
following_url	A	https://api.github.com/users/achiu/following
gists_url	A	https://api.github.com/users/achiu/gists
starred_url	A	https://api.github.com/users/achiu/starred
subscriptions_url	A	https://api.github.com/users/achiu/subscriptions
organizations_url	A	https://api.github.com/users/achiu/orgs
repos_url	A	https://api.github.com/users/achiu/repos
events_url	A	https://api.github.com/users/achiu/events
received_events_url	A	https://api.github.com/users/achiu/received_events
type	A	User
site_admin	B1	true

Cookies Automatically Send Cookies
 Store Received Cookies

Authorization Set HTTP Basic Auth
Set OAuth 1 Set OAuth 2
Set Amazon S3 Signature

Swift (Alamofire) Copy Export Open With

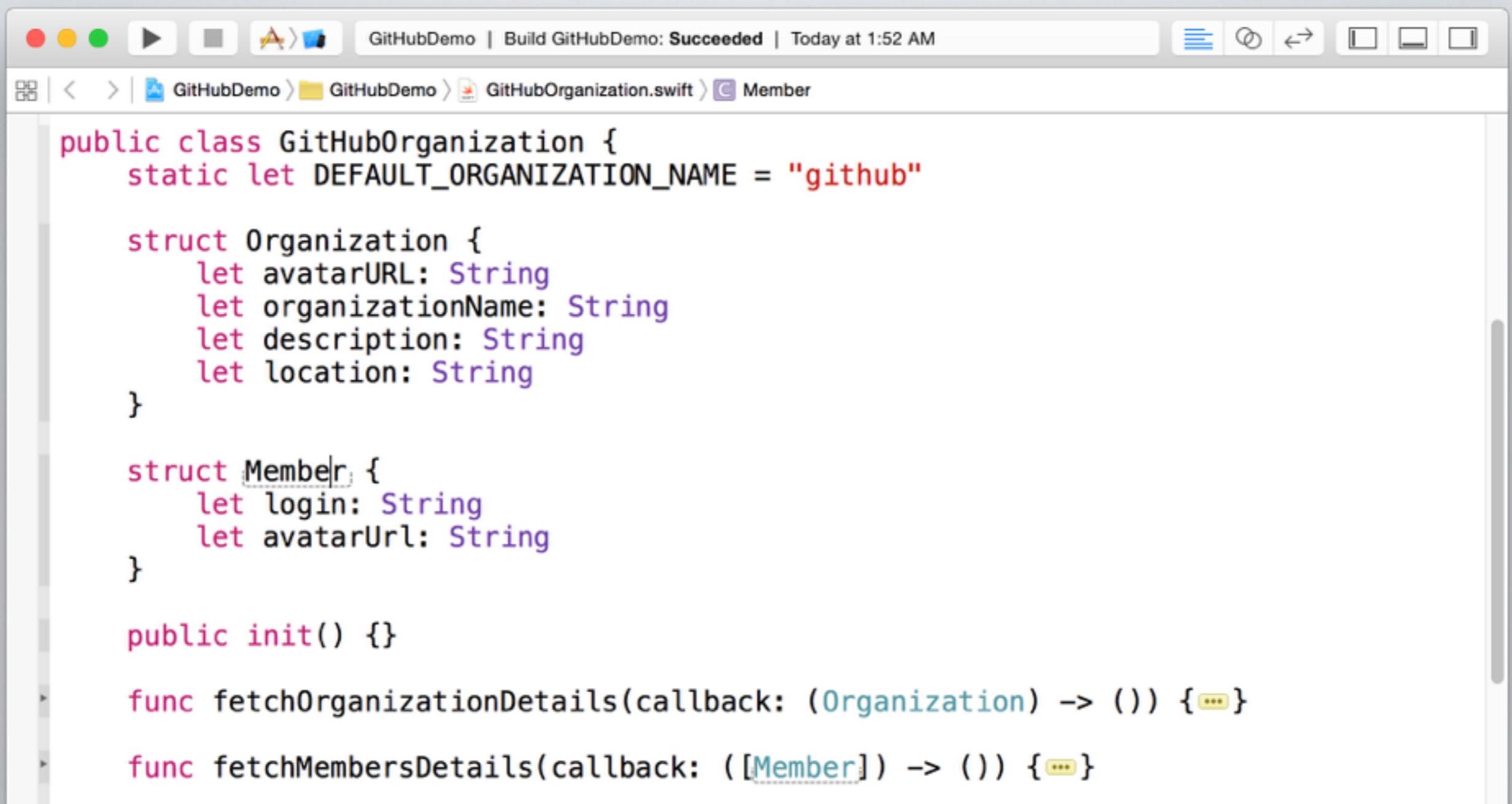


SWIFTYJSON

SWIFTYJSON

```
1. vazexqi@daigo: ~/Development/TalentSpark/GitHubDemo (zsh)
vazexqi@daigo:~/Development/TalentSpark/GitHubDemo|→ pod install| ± master ✓ 9:52:35
```

MODEL



A screenshot of an Xcode workspace window. The title bar shows "GitHubDemo | Build GitHubDemo: Succeeded | Today at 1:52 AM". The file path in the navigation bar is "GitHubDemo > GitHubDemo > GitHubOrganization.swift". The main editor area contains the following Swift code:

```
public class GitHubOrganization {
    static let DEFAULT_ORGANIZATION_NAME = "github"

    struct Organization {
        let avatarURL: String
        let organizationName: String
        let description: String
        let location: String
    }

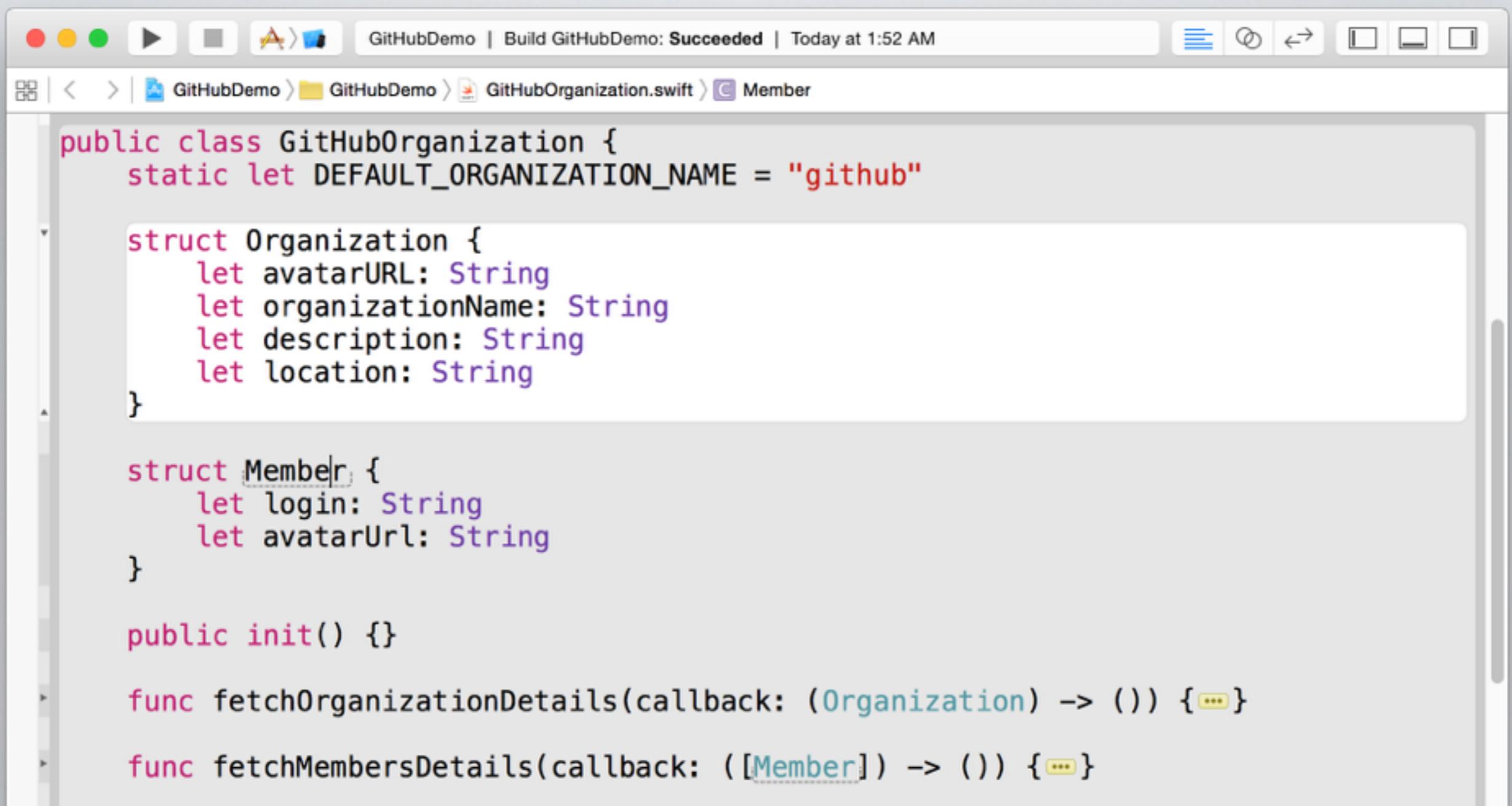
    struct Member {
        let login: String
        let avatarUrl: String
    }

    public init() {}

    func fetchOrganizationDetails(callback: (Organization) -> ()) { ... }

    func fetchMembersDetails(callback: ([Member]) -> ()) { ... }
}
```

MODEL



A screenshot of an Xcode workspace window. The title bar shows "GitHubDemo | Build GitHubDemo: Succeeded | Today at 1:52 AM". The file path in the navigation bar is "GitHubDemo > GitHubDemo > GitHubOrganization.swift". The main editor area contains the following Swift code:

```
public class GitHubOrganization {
    static let DEFAULT_ORGANIZATION_NAME = "github"

    struct Organization {
        let avatarURL: String
        let organizationName: String
        let description: String
        let location: String
    }

    struct Member {
        let login: String
        let avatarUrl: String
    }

    public init() {}

    func fetchOrganizationDetails(callback: (Organization) -> ()) { ... }

    func fetchMembersDetails(callback: ([Member]) -> ()) { ... }
}
```

MODEL



The screenshot shows the Xcode interface with the file `GitHubOrganization.swift` open. The code defines a class `GitHubOrganization` with static properties and two nested structures: `Organization` and `Member`. The `Organization` structure contains properties for avatar URL, organization name, description, and location. The `Member` structure contains properties for login and avatar URL. The code also includes `init()`, `fetchOrganizationDetails`, and `fetchMembersDetails` methods.

```
public class GitHubOrganization {
    static let DEFAULT_ORGANIZATION_NAME = "github"

    struct Organization {
        let avatarURL: String
        let organizationName: String
        let description: String
        let location: String
    }

    struct Member {
        let login: String
        let avatarUrl: String
    }

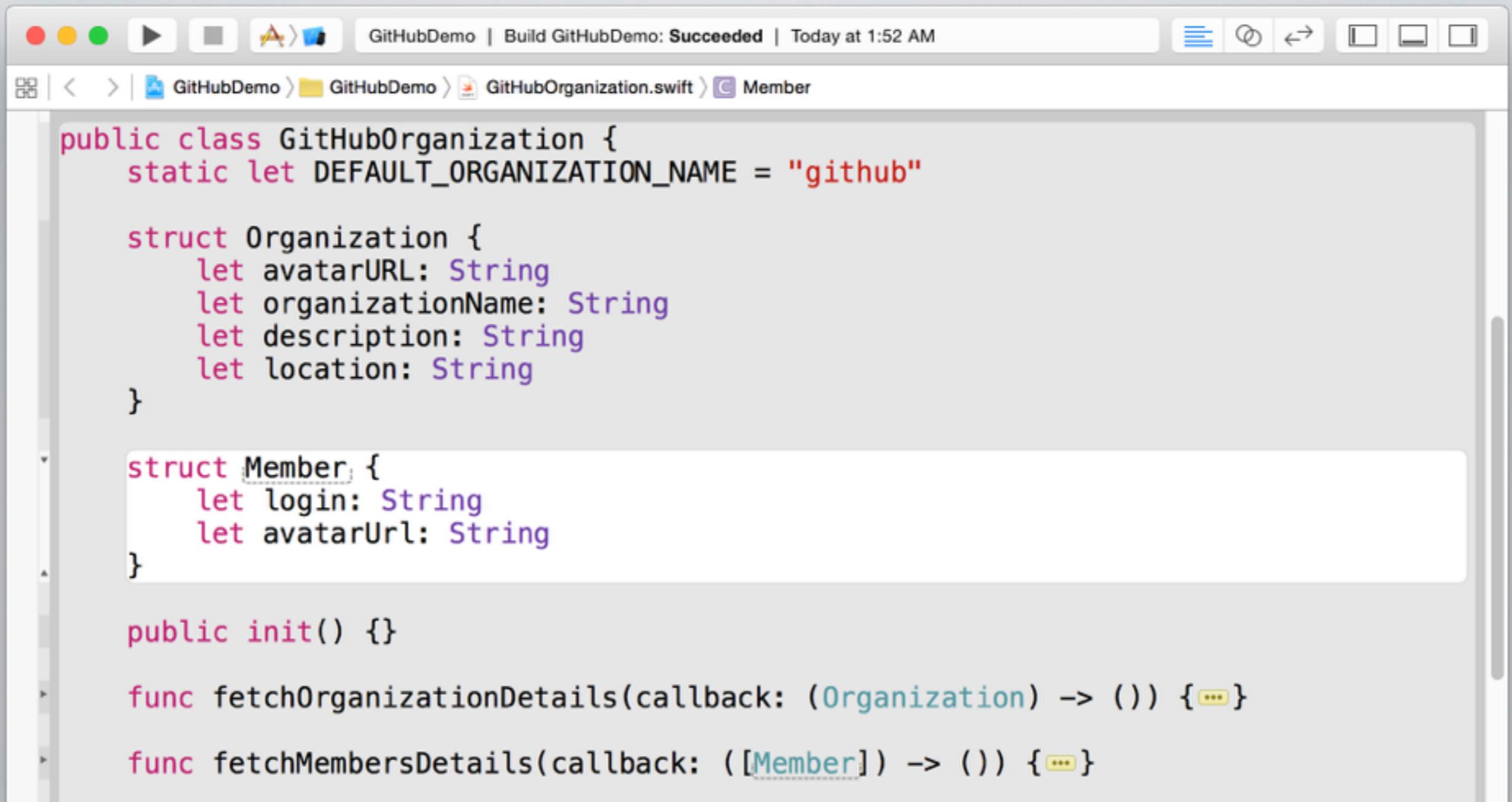
    public init() {}

    func fetchOrganizationDetails(callback: (Organization) -> ()) { ... }

    func fetchMembersDetails(callback: ([Member]) -> ()) { ... }
}
```

Load the icon, organization name, description and location

MODEL



A screenshot of an Xcode workspace window. The title bar shows "GitHubDemo | Build GitHubDemo: Succeeded | Today at 1:52 AM". The file path in the navigation bar is "GitHubDemo > GitHubDemo > GitHubOrganization.swift > Member". The main editor area contains the following Swift code:

```
public class GitHubOrganization {
    static let DEFAULT_ORGANIZATION_NAME = "github"

    struct Organization {
        let avatarURL: String
        let organizationName: String
        let description: String
        let location: String
    }

    struct Member {
        let login: String
        let avatarUrl: String
    }

    public init() {}

    func fetchOrganizationDetails(callback: (Organization) -> ()) { ... }

    func fetchMembersDetails(callback: ([Member]) -> ()) { ... }
}
```

MODEL



The screenshot shows the Xcode interface with the title bar "GitHubDemo | Build GitHubDemo: Succeeded | Today at 1:52 AM". The navigation bar shows the file path "GitHubDemo > GitHubDemo > GitHubOrganization.swift > Member". The main editor area contains the following Swift code:

```
public class GitHubOrganization {
    static let DEFAULT_ORGANIZATION_NAME = "github"

    struct Organization {
        let avatarURL: String
        let organizationName: String
        let description: String
        let location: String
    }

    struct Member {
        let login: String
        let avatarUrl: String
    }

    public init() {}

    func fetchOrganizationDetails(callback: (Organization) -> ()) { ... }

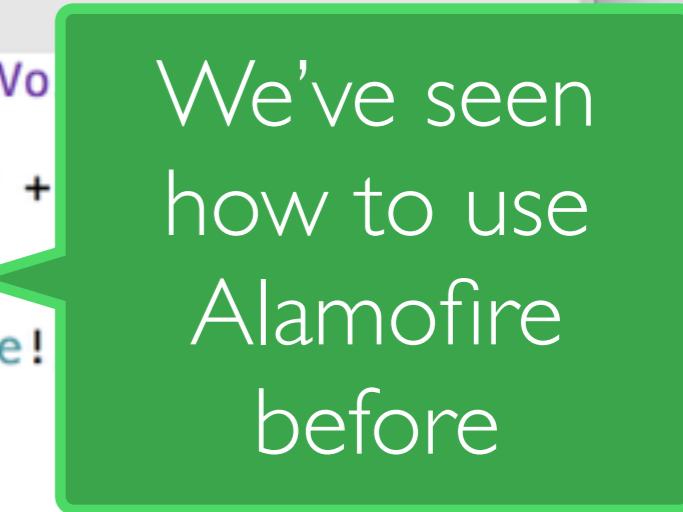
    func fetchMembersDetails(callback: ([Member]) -> ()) { ... }
}
```

A green callout bubble with a white arrow points from the bottom right towards the "Member" struct definition. Inside the bubble, the text "Just need the icon and username" is displayed.

MODEL

```
26 }
27
28 func fetchOrganizationDetails(callback: (Organization) -> Void) {
29     // Fetch organization details
30     Alamofire.request(.GET, "https://api.github.com/orgs/" +
31         GitHubOrganization.DEFAULT_ORGANIZATION_NAME)
32     .responseJSON { _, _, jsonObj in
33         self.populateOrganizationInfoWith(jsonObj.value!, callback:
34             callback)
35     }
36
37     func populateOrganizationInfoWith(data: AnyObject?, callback: (Organization) ->
38         Void) {
39         let json = JSON(data!)
40         callback(Organization(avatarURL: json["avatar_url"].stringValue,
41             organizationName: json["name"].stringValue, description:
42                 json["description"].stringValue, location:
43                     json["location"].stringValue))
44     }
45
46     func fetchMembersDetails(callback: ([Member]) -> Void) {
47         // Fetch organization members
48         Alamofire.request(.GET, "https://api.github.com/orgs/" +
49             GitHubOrganization.DEFAULT_ORGANIZATION_NAME + "/members")
```

MODEL



```
GitHubOrganization.swift
GitHubDemo > GitHubDemo > GitHubOrganization.swift > GitHubOrganization

26 }
27
28 func fetchOrganizationDetails(callback: (Organization) -> Void) {
29     // Fetch organization details
30     Alamofire.request(.GET, "https://api.github.com/orgs/" +
31         GitHubOrganization.DEFAULT_ORGANIZATION_NAME)
32         .responseJSON { _, _, jsonObj in
33             self.populateOrganizationInfoWith(jsonObj.value!
34                 callback)
35         }
36
37     func populateOrganizationInfoWith(data: AnyObject?, callback: (Organization) ->
38         Void) {
39         let json = JSON(data!)
40         callback(Organization(avatarURL: json["avatar_url"].stringValue,
41             organizationName: json["name"].stringValue, description:
42                 json["description"].stringValue, location:
43                 json["location"].stringValue))
44     }
45
46     func fetchMembersDetails(callback: ([Member]) -> Void) {
47         // Fetch organization members
48         Alamofire.request(.GET, "https://api.github.com/orgs/" +
49             GitHubOrganization.DEFAULT_ORGANIZATION_NAME + "/members")
```

We've seen
how to use
Alamofire
before

MODEL

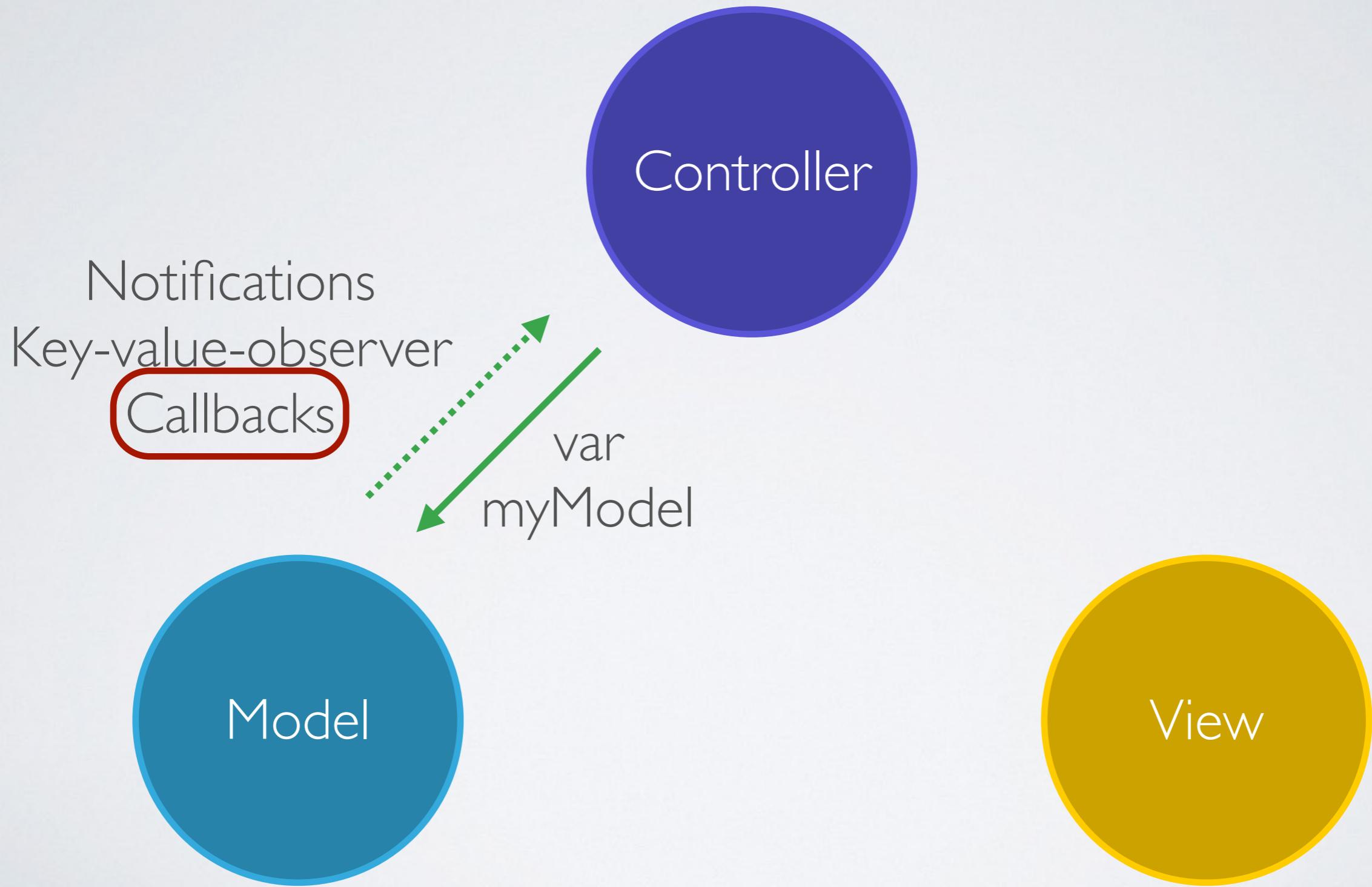
```
GitHubOrganization.swift
GitHubDemo > GitHubDemo > GitHubOrganization.swift > GitHubOrganization

26 }
27
28 func fetchOrganizationDetails(callback: (Organization) -> Void) {
29     // Fetch organization details
30     Alamofire.request(.GET, "https://api.github.com/orgs/" +
31         GitHubOrganization.DEFAULT_ORGANIZATION_NAME)
32     .responseJSON { _, _, jsonObj in
33         self.populateOrganizationInfoWith(jsonObj.value!
34             callback)
35     }
36
37     func populateOrganizationInfoWith(data: AnyObject?, callback: (Organization) ->
38         Void) {
39         let json = JSON(data!)
40         callback(Organization(avatarURL: json["avatar_url"].stringValue,
41             name: json["name"].stringValue, description:
42                 json["description"].stringValue, location:
43                     json["location"].stringValue))
44     }
45
46     func fetchOrganizationMembers(callback: ([Member]) -> Void) {
47         // Fetch organization members
48         Alamofire.request(.GET, "https://api.github.com/orgs/" +
49             GitHubOrganization.DEFAULT_ORGANIZATION_NAME + "/members")
```

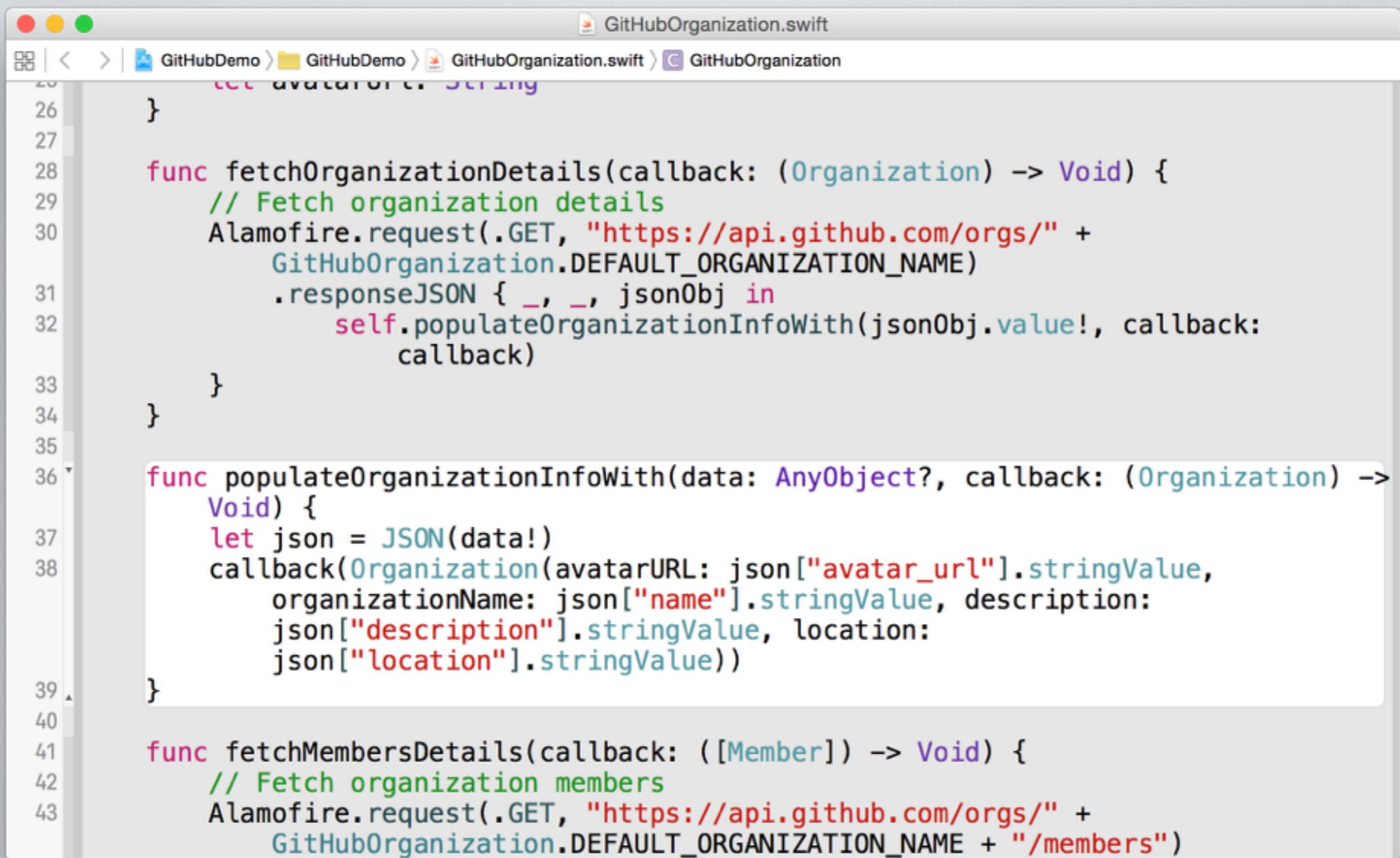
This is
SwiftyJSON

We've seen
how to use
Alamofire
before

NON-SPAGHETTI CODE



MODEL



The screenshot shows a Mac OS X desktop with an Xcode project window open. The title bar says "GitHubOrganization.swift". The file path in the top left is "GitHubDemo > GitHubDemo > GitHubOrganization.swift". The main area displays Swift code for a GitHub organization model. The code includes functions for fetching organization details, populating organization info from JSON, and fetching member details using Alamofire.

```
GitHubOrganization.swift
GitHubDemo > GitHubDemo > GitHubOrganization.swift > GitHubOrganization

26 }
27
28 func fetchOrganizationDetails(callback: (Organization) -> Void) {
29     // Fetch organization details
30     Alamofire.request(.GET, "https://api.github.com/orgs/" +
31         GitHubOrganization.DEFAULT_ORGANIZATION_NAME)
32         .responseJSON { _, _, jsonObj in
33             self.populateOrganizationInfoWith(jsonObj.value!, callback:
34                 callback)
35         }
36
37     func populateOrganizationInfoWith(data: AnyObject?, callback: (Organization) ->
38         Void) {
39         let json = JSON(data!)
40         callback(Organization(avatarURL: json["avatar_url"].stringValue,
41             organizationName: json["name"].stringValue, description:
42                 json["description"].stringValue, location:
43                     json["location"].stringValue))
44
45     func fetchMembersDetails(callback: ([Member]) -> Void) {
46         // Fetch organization members
47         Alamofire.request(.GET, "https://api.github.com/orgs/" +
48             GitHubOrganization.DEFAULT_ORGANIZATION_NAME + "/members")
```

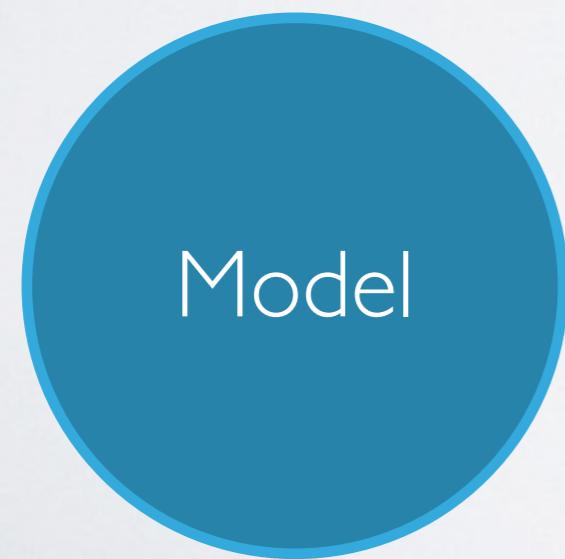
MODEL

```
GitHubOrganization.swift
GitHubDemo > GitHubDemo > GitHubOrganization.swift > GitHubOrganization

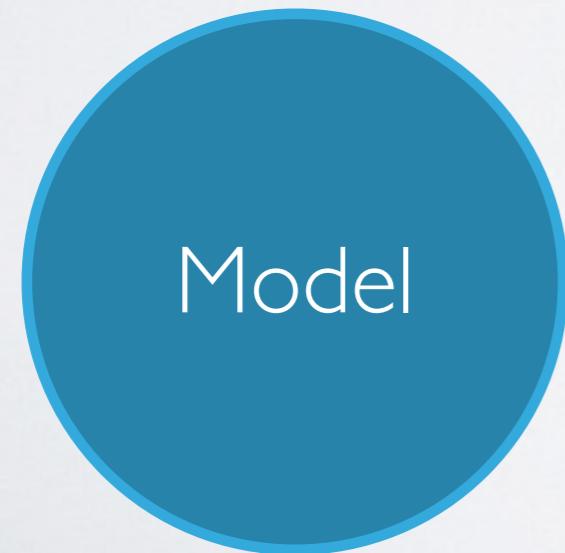
26 }
27
28 func fetchOrganizationDetails(callback: (Organization) -> Void) {
29     // Fetch organization details
30     Alamofire.request(.GET, "https://api.github.com/orgs/" +
31         GitHubOrganization.DEFAULT_ORGANIZATION_NAME)
32         .responseJSON { _, _, jsonObj in
33             self.populateOrganizationInfoWith(jsonObj.value!, callback:
34                 callback)
35         }
36
37     func populateOrganizationInfoWith(data: AnyObject?, callback: (Organization) ->
38         Void) {
39         let json = JSON(data!)
40         callback(Organization(avatarURL: json["avatar_url"].stringValue,
41             organizationName: json["name"].str
42             json["description"].stringValue, l
43             json["location"].stringValue))
44
45     func fetchMembersDetails(callback: ([Member] -> Void) {
46         // Fetch organization members
47         Alamofire.request(.GET, "https://api.g
48             GitHubOrganization.DEFAULT_ORGANIZA
```

How we
communicate
with our
controller

CONTROLLER



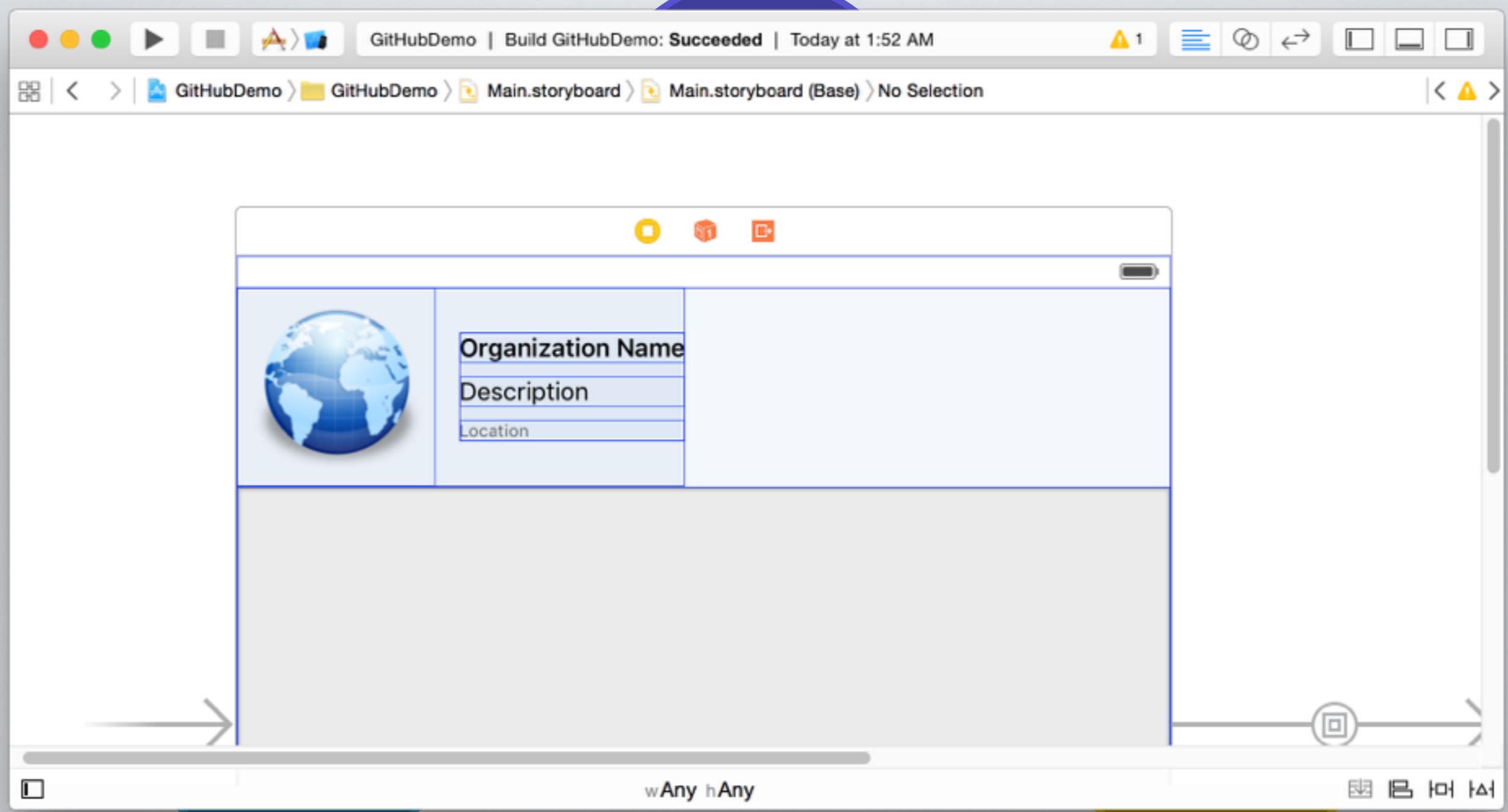
CONTROLLER



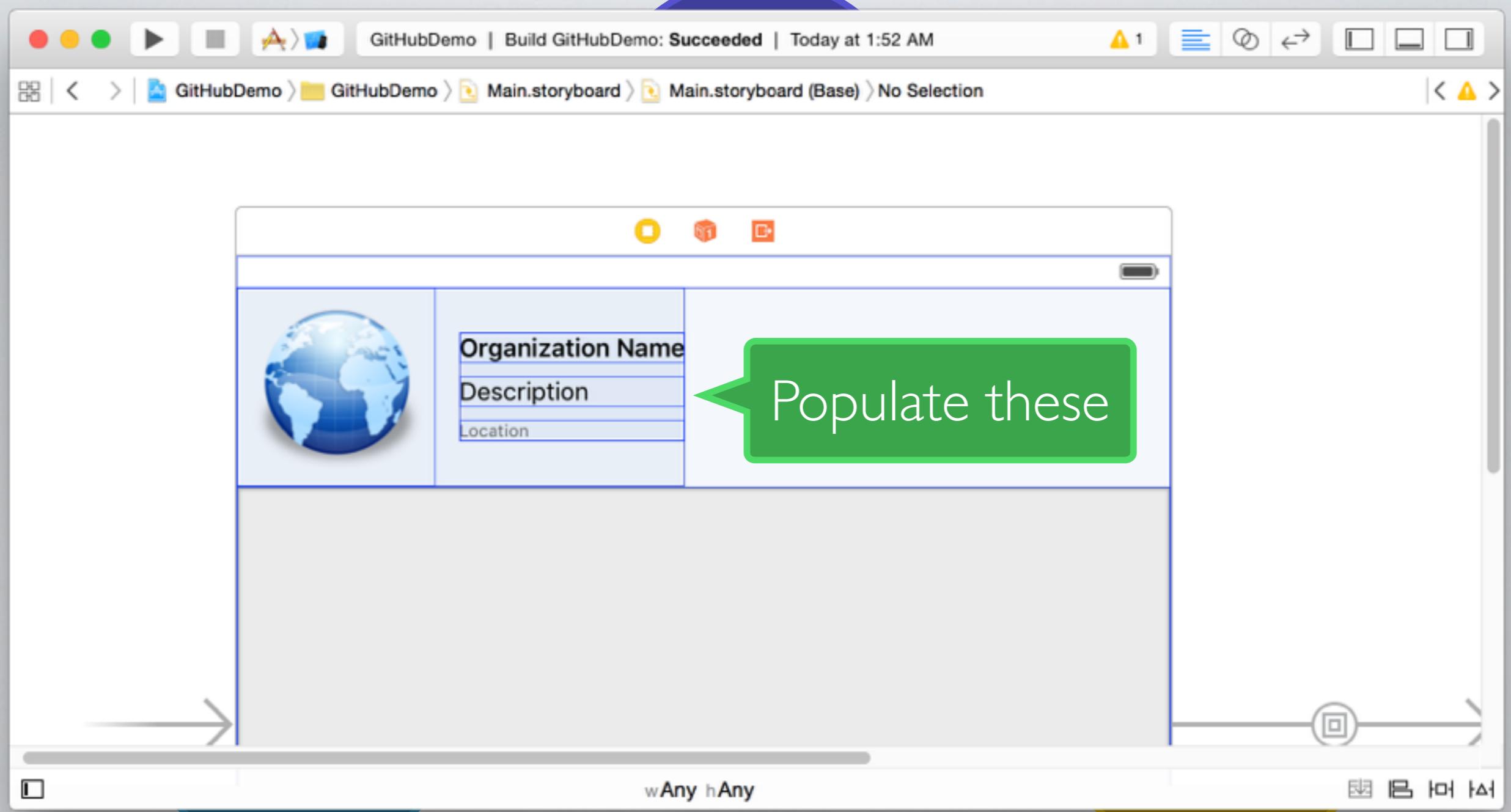
CONTROLLER #1



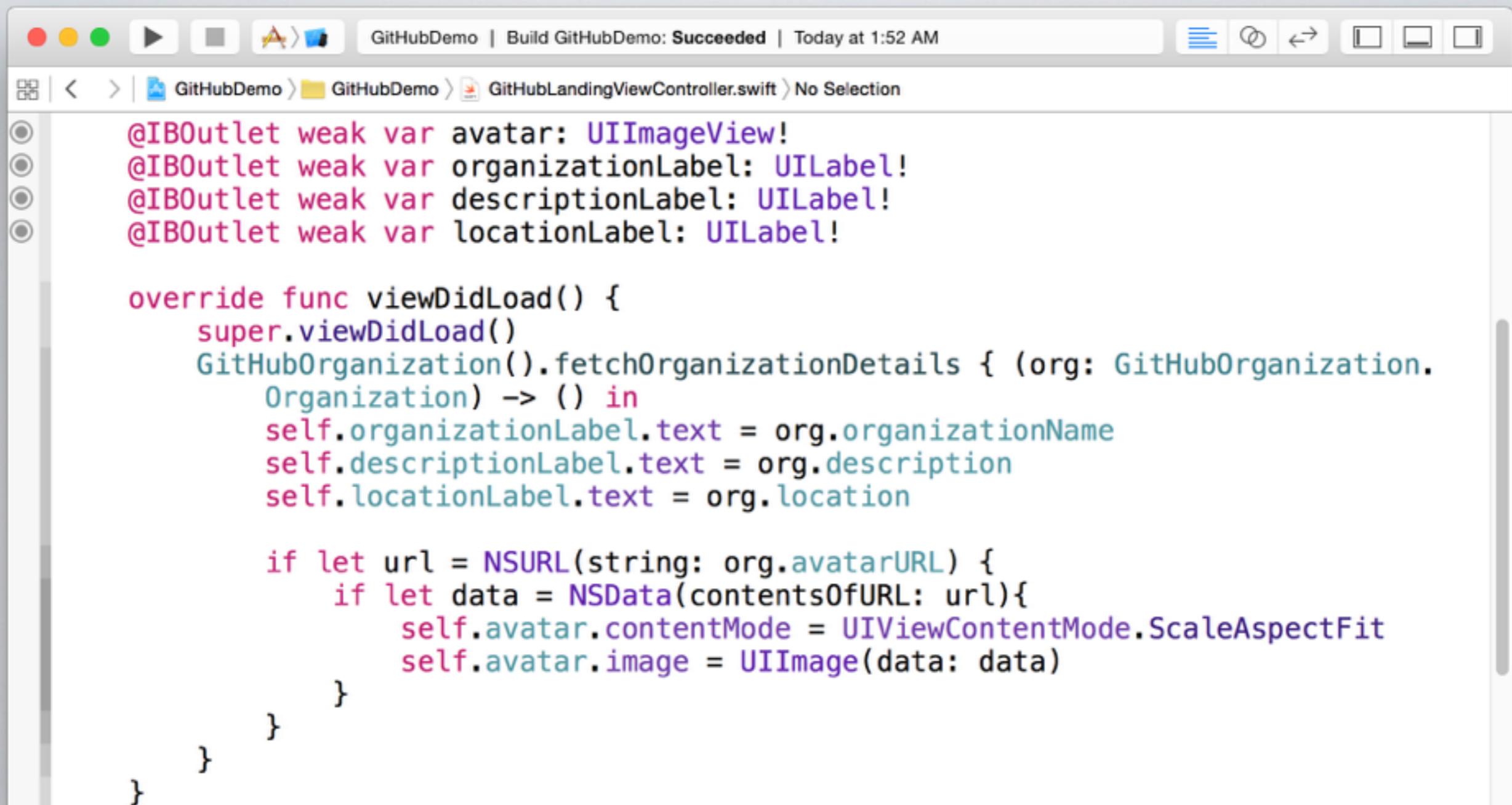
CONTROLLER #1



CONTROLLER #1



CONTROLLER #1



The screenshot shows the Xcode interface with the following details:

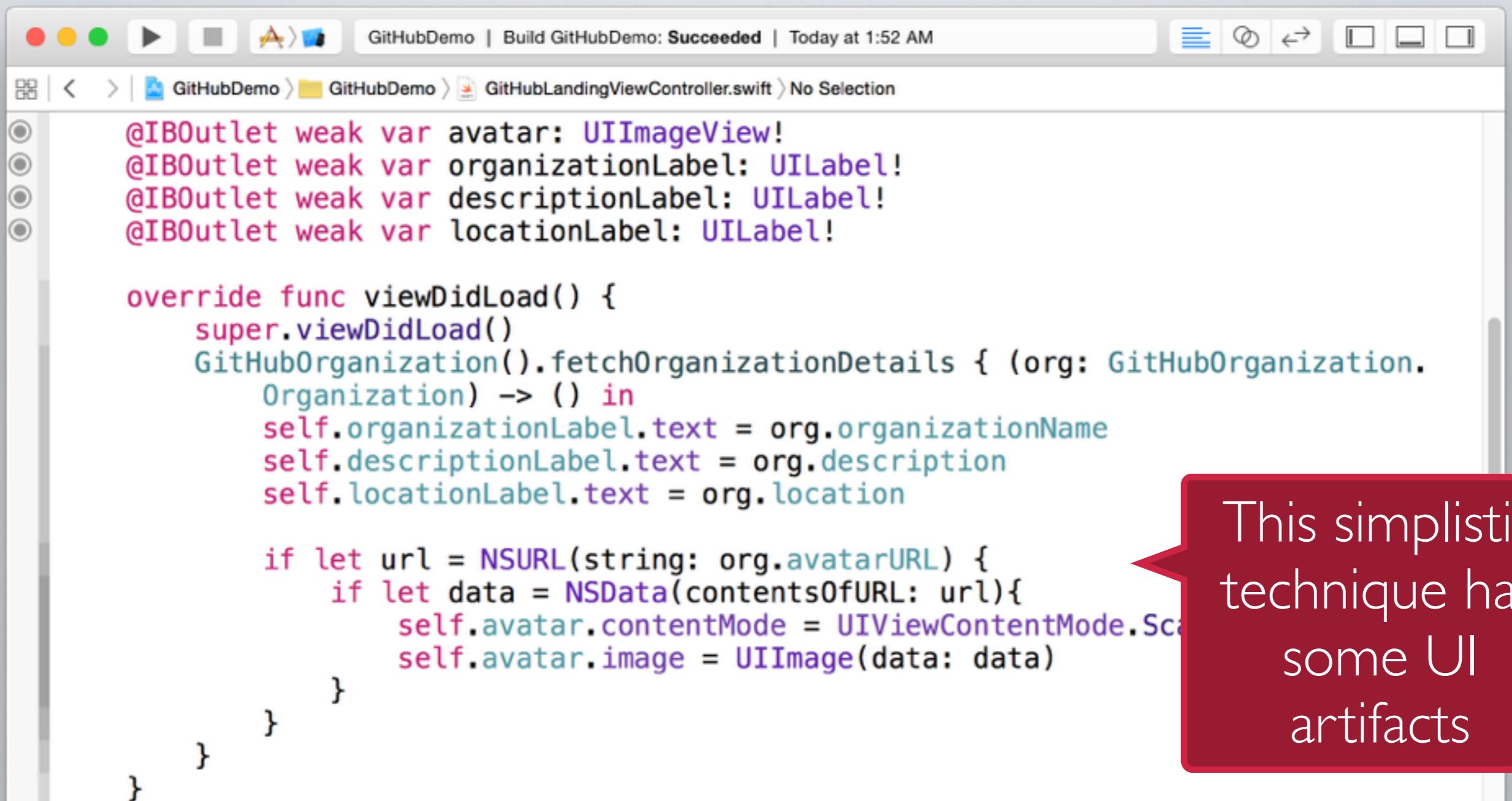
- Toolbar:** Standard Xcode toolbar with icons for file, edit, search, and build.
- Status Bar:** GitHubDemo | Build GitHubDemo: Succeeded | Today at 1:52 AM
- Project Navigator:** Shows the project structure: GitHubDemo > GitHubDemo > GitHubLandingViewController.swift
- Code Editor:** Displays the `GitHubLandingViewController.swift` file content. The code is written in Swift and defines a view controller for displaying organization details.

```
    @IBOutlet weak var avatar: UIImageView!
    @IBOutlet weak var organizationLabel: UILabel!
    @IBOutlet weak var descriptionLabel: UILabel!
    @IBOutlet weak var locationLabel: UILabel!

    override func viewDidLoad() {
        super.viewDidLoad()
        GitHubOrganization().fetchOrganizationDetails { (org: GitHubOrganization.Organizations) -> () in
            self.organizationLabel.text = org.organizationName
            self.descriptionLabel.text = org.description
            self.locationLabel.text = org.location

            if let url = NSURL(string: org.avatarURL) {
                if let data = NSData(contentsOfURL: url){
                    self.avatar.contentMode = UIViewContentMode.ScaleAspectFit
                    self.avatar.image = UIImage(data: data)
                }
            }
        }
    }
```

CONTROLLER #1



The screenshot shows the Xcode interface with the title bar "GitHubDemo | Build GitHubDemo: Succeeded | Today at 1:52 AM". The file path in the navigation bar is "GitHubDemo > GitHubDemo > GitHubLandingViewController.swift". The code editor contains the following Swift code:

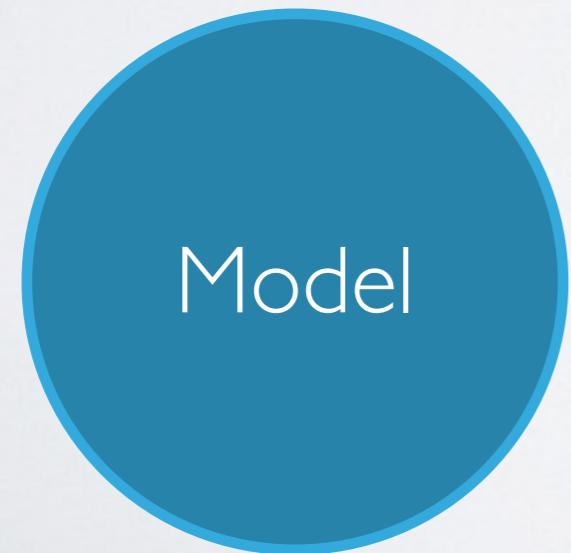
```
    @IBOutlet weak var avatar: UIImageView!
    @IBOutlet weak var organizationLabel: UILabel!
    @IBOutlet weak var descriptionLabel: UILabel!
    @IBOutlet weak var locationLabel: UILabel!

    override func viewDidLoad() {
        super.viewDidLoad()
        GitHubOrganization().fetchOrganizationDetails { (org: GitHubOrganization.Organizations) -> () in
            self.organizationLabel.text = org.organizationName
            self.descriptionLabel.text = org.description
            self.locationLabel.text = org.location

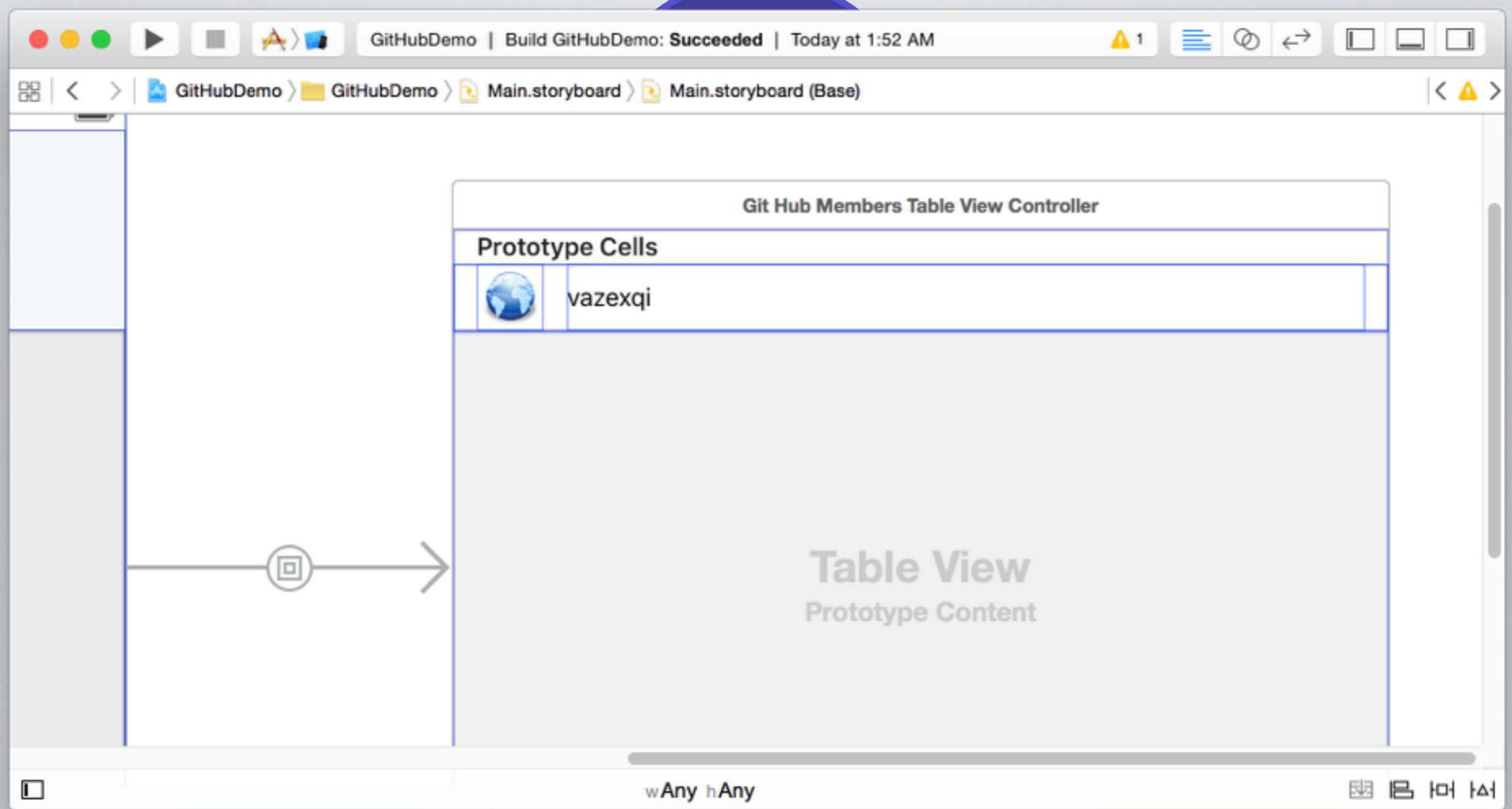
            if let url = NSURL(string: org.avatarURL) {
                if let data = NSData(contentsOfURL: url){
                    self.avatar.contentMode = UIViewContentMode.ScaleAspectFit
                    self.avatar.image = UIImage(data: data)
                }
            }
        }
    }
```

A red callout bubble with a white border and a black arrow points from the bottom right towards the code. Inside the bubble, the text reads: "This simplistic technique has some UI artifacts".

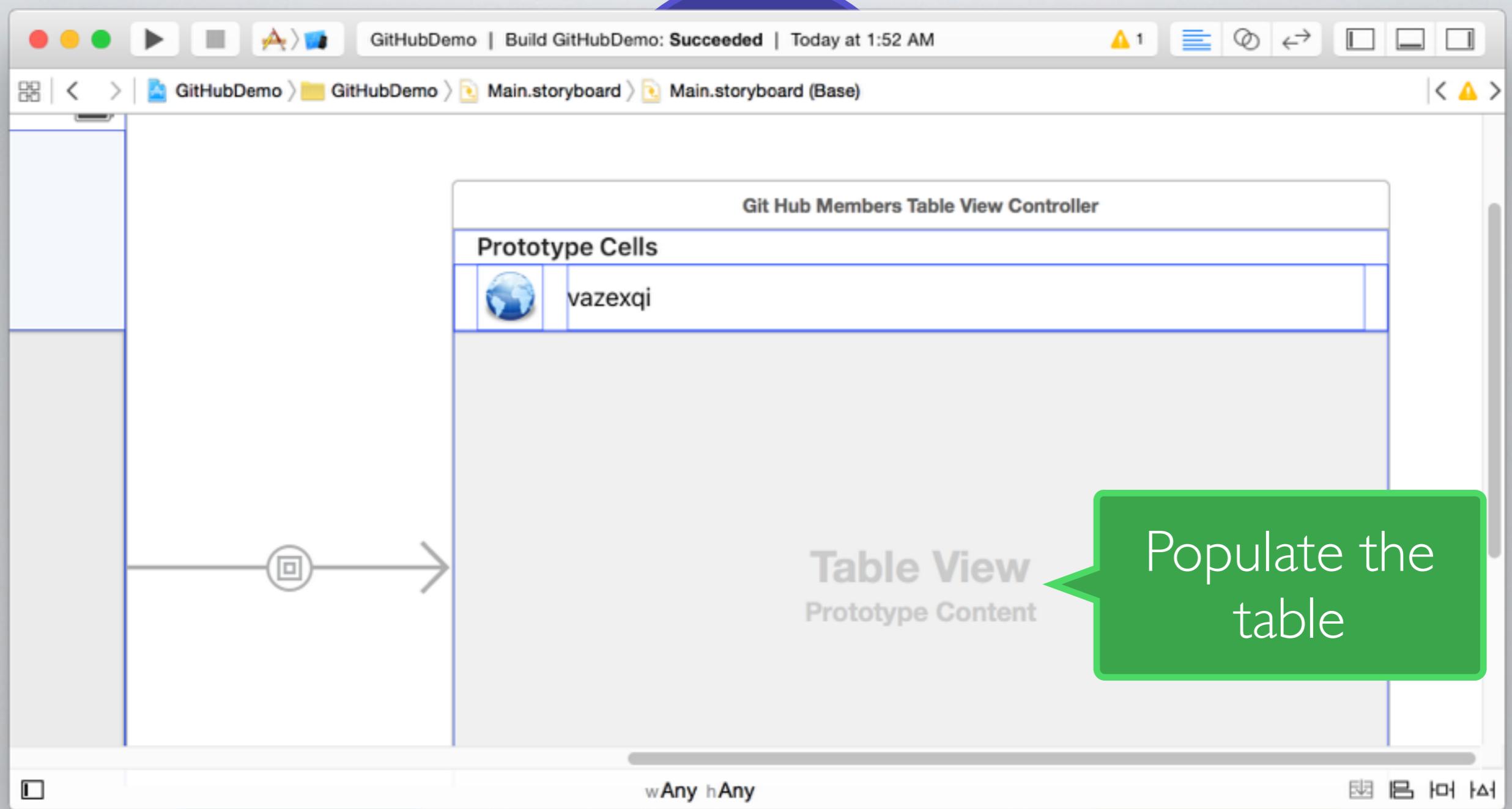
CONTROLLER #2



CONTROLLER #2



CONTROLLER #2



TABLEVIEWCONTROLLER

UITableViewController Class Reference

inherits From: NSObject, UIResponder, UIViewController, UITableViewController, CABTMIDICentralViewC...

Conforms To: UIResponder, NSObject, UIAppearanceContainer, UITraitEnvironment, UITableViewDataSource, UITableViewDelegate

Import Statement: `OBJECTION-C @import UIKit;`

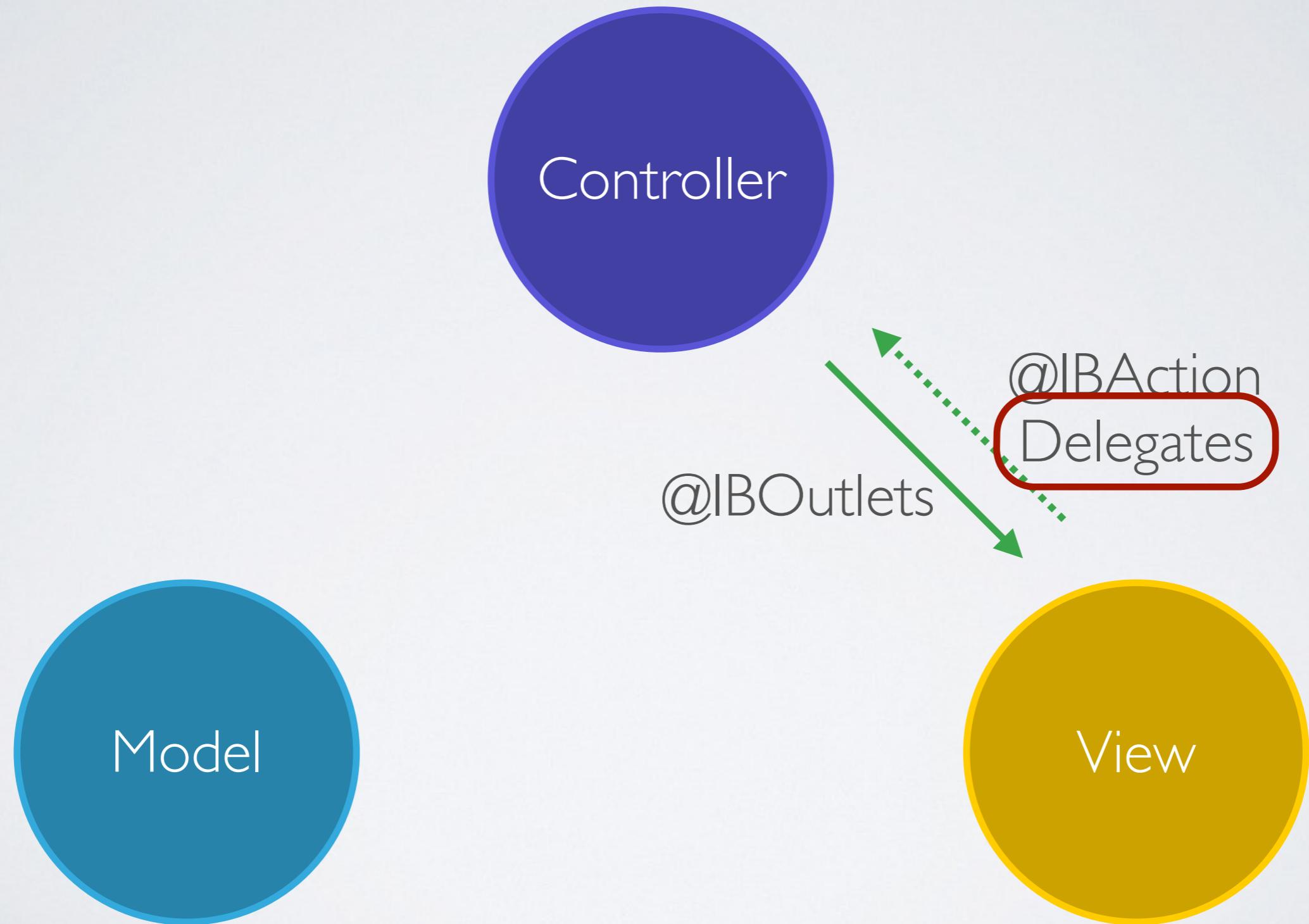
Availability: Available in iOS 2.0 and later

The `UITableViewController` class creates a controller object that manages a table view. It implements the following behavior:

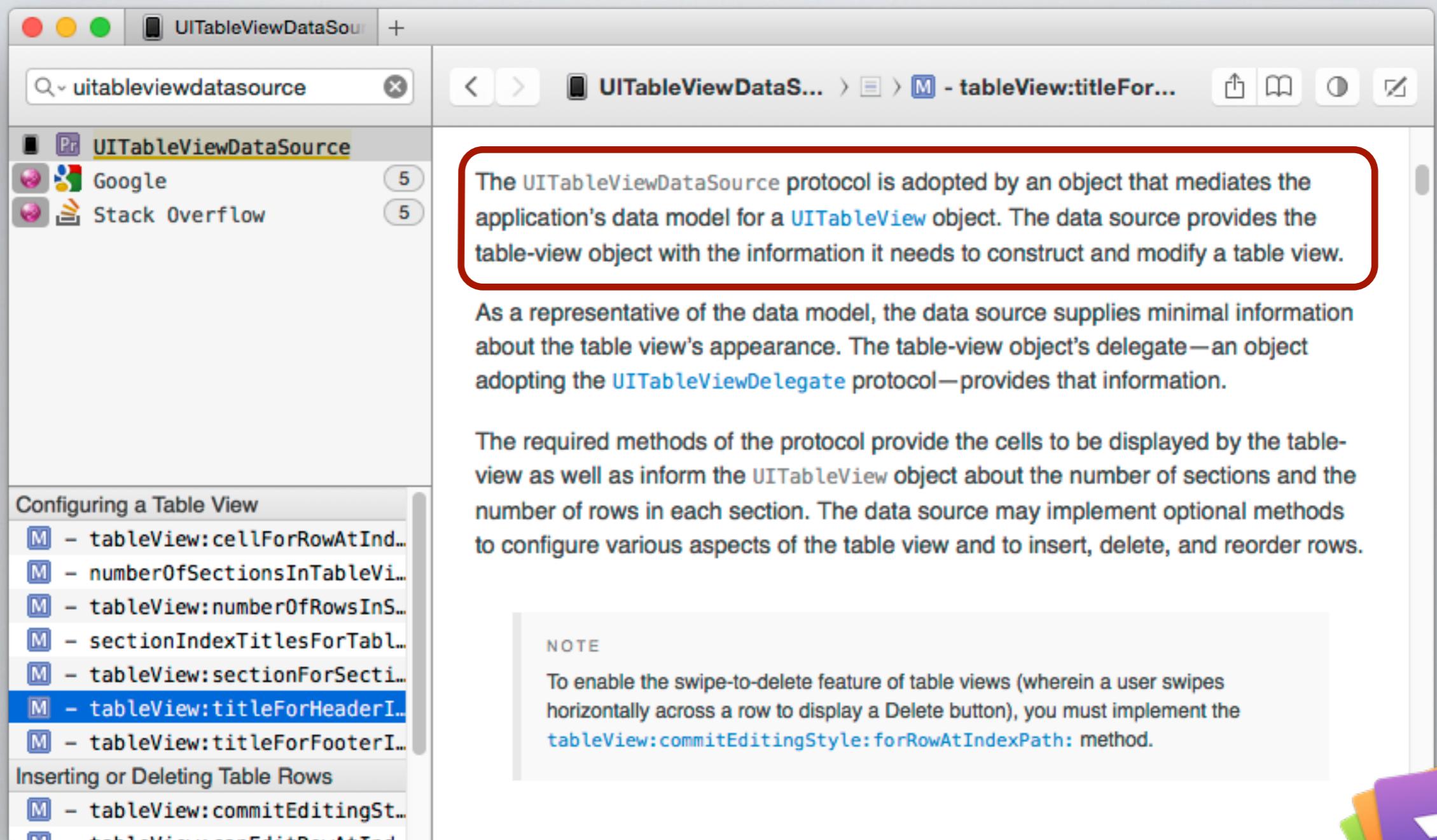
- If a nib file is specified via the `initWithNibName:bundle:` method (which is declared by the superclass `UIViewController`), `UITableViewController` loads the table view archived in the nib file. Otherwise, it creates an unconfigured...



NON-SPAGHETTI CODE



UITABLEVIEWDATASOURCE



The screenshot shows the Apple Developer Documentation for the `UITableViewDataSource` protocol. The search bar at the top contains the text "uitableviewdatasource". The main content area shows the `UITableViewController` class with the `- tableView:titleFor...` method selected. A red box highlights the following text in the description:

The `UITableViewDataSource` protocol is adopted by an object that mediates the application's data model for a `UITableView` object. The data source provides the table-view object with the information it needs to construct and modify a table view.

Below this, another paragraph explains the role of the data source:

As a representative of the data model, the data source supplies minimal information about the table view's appearance. The table-view object's delegate—an object adopting the `UITableViewDelegate` protocol—provides that information.

A sidebar on the left lists sections such as "Configuring a Table View" and "Inserting or Deleting Table Rows", each with a list of methods. The "Configuring a Table View" section includes the `- tableView:titleForHeaderI...` method, which is highlighted with a blue box. A "NOTE" section at the bottom right provides instructions for enabling swipe-to-delete functionality.

Configuring a Table View

- M - `tableView:cellForRowAtIndex...`
- M - `numberOfSectionsInTableVi...`
- M - `tableView:numberOfRowsInS...`
- M - `sectionIndexTitlesForTabl...`
- M - `tableView:sectionForSecti...`
- M - `tableView:titleForHeaderI...`
- M - `tableView:titleForFooterI...`

Inserting or Deleting Table Rows

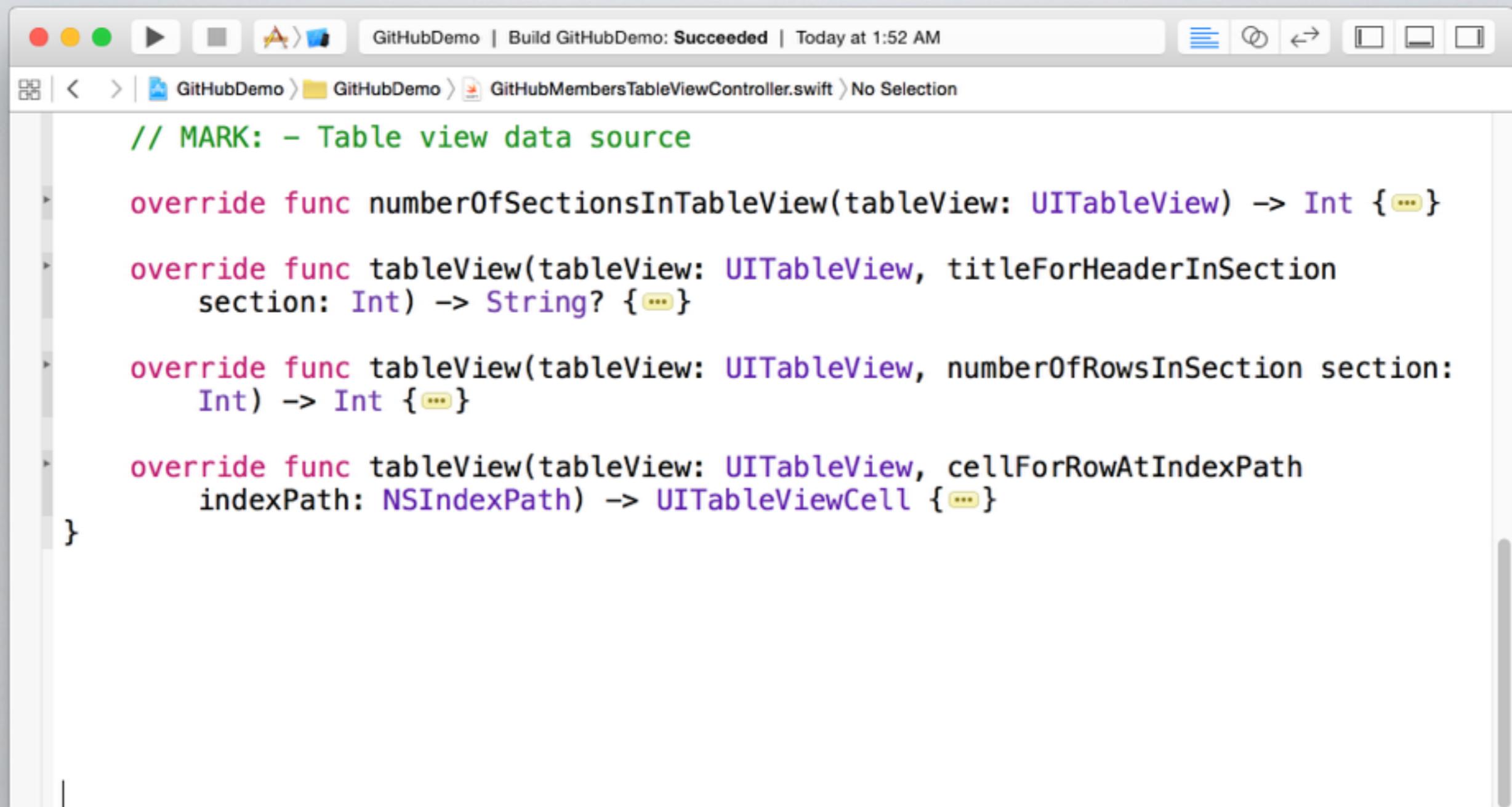
- M - `tableView:commitEditingStyle...`
- M - `tableView:canEditRowAtIndexPath...`

NOTE

To enable the swipe-to-delete feature of table views (wherein a user swipes horizontally across a row to display a Delete button), you must implement the `tableView:commitEditingStyle:forRowAtIndexPath:` method.



UITABLEVIEWDATASOURCE



The screenshot shows a Xcode interface with the following details:

- Toolbar:** Standard Xcode toolbar with icons for file operations.
- Status Bar:** Displays "GitHubDemo | Build GitHubDemo: Succeeded | Today at 1:52 AM".
- Project Navigator:** Shows the project structure: GitHubDemo > GitHubDemo > GitHubMembersTableViewController.swift.
- Code Editor:** Displays Swift code for a UITableViewDataSource implementation. The code includes four overridden methods: `numberOfSectionsInTableView`, `titleForHeaderInSection`, `numberOfRowsInSection`, and `cellForRowAtIndexPath`. A closing brace at the end of the class definition indicates the implementation of the UITableViewDataSource protocol.

```
// MARK: - Table view data source

override func numberOfSectionsInTableView(tableView: UITableView) -> Int { ... }

override func tableView(tableView: UITableView, titleForHeaderInSection section: Int) -> String? { ... }

override func tableView(tableView: UITableView, numberOfRowsInSection section: Int) -> Int { ... }

override func tableView(tableView: UITableView, cellForRowAt indexPath: NSIndexPath) -> UITableViewCell { ... }

}
```

UITABLEVIEWDATASOURCE

The screenshot shows a Xcode interface with the title bar "GitHubDemo | Build GitHubDemo: Succeeded | Today at 1:52 AM". The navigation bar shows the file path "GitHubDemo > GitHubDemo > GitHubMembersTableViewController.swift" and the selected method "numberOfSectionsInTableView(_:)". The code editor contains the following Swift code:

```
// MARK: - Table view data source

override func numberOfSectionsInTableView(tableView: UITableView) -> Int { ... }
```

A callout box provides detailed documentation for the `numberOfSectionsInTableView` method:

- Declaration:** `override func numberOfSectionsInTableView(tableView: UITableView) -> Int`
- Description:** Asks the data source to return the number of sections in the table view.
- Parameters:** `tableView` An object representing the table view requesting this information.
- Returns:** The number of sections in `tableView`.
- Availability:** iOS (8.1 and later)
- Declared In:** UIKit
- Reference:** [UITableViewDataSource Protocol Reference](#)

The code editor also shows partial implementations for other UITableViewDataSource methods:

- `w, titleForHeaderInSection`
- `w, numberOfRowsInSection section:`
- `w, cellForRowAt indexPath`
- `l { ... }`

UITABLEVIEWDATASOURCE

The screenshot shows a Xcode interface with the title bar "GitHubDemo | Build GitHubDemo: Succeeded | Today at 1:52 AM". The navigation bar shows the file path "GitHubDemo > GitHubDemo > GitHubMembersTableViewController.swift" and the selected method "tableView(_:titleForHeaderInSection:)". The code editor contains the following Swift code:

```
// MARK: - Table view data source

override func numberOfSectionsInTableView(tableView: UITableView) -> Int { ... }

override func tableView(tableView: UITableView, titleForHeaderInSection section: Int) -> String?
```

A callout bubble provides detailed documentation for the `tableView(_:titleForHeaderInSection:)` method:

- Declaration:** `override func tableView(tableView: UITableView, titleForHeaderInSection section: Int) -> String?`
- Description:** Asks the data source for the title of the header of the specified section of the table view.
- Parameters:**
 - `tableView`: The table-view object asking for the title.
 - `section`: An index number identifying a section of `tableView`.
- Returns:** A string to use as the title of the section header.
- Availability:** iOS (8.1 and later)
- Related Declarations:**
 - `tableView(_:numberOfRowsInSection:)`
 - `tableView(_:cellForRowAtIndexPath:)`
 - `tableView`
 - [Show more...](#)
- Declared In:** UIKit
- Reference:** [UITableViewDataSource Protocol Reference](#)

UITABLEVIEWDATASOURCE

The screenshot shows an Xcode interface with a file named `GitHubMembersTableViewController.swift` open. The code is implementing the `UITableViewDataSource` protocol. A tooltip is displayed over the `override func tableView(tableView: UITableView, numberOfRowsInSection section: Int)` method, providing documentation and details about its parameters and return value.

```
// MARK: - Table view data source

override func numberOfSectionsInTableView(tableView: UITableView) -> Int { ... }

override func tableView(tableView: UITableView, titleForHeaderInSection section: Int) -> String? { ... }

override func tableView(tableView: UITableView, numberOfRowsInSection section:
```

Declaration: `override func tableView(tableView: UITableView, numberOfRowsInSection section: Int) -> Int`

Description: Tells the data source to return the number of rows in a given section of a table view.

Parameters: `tableView` The table-view object requesting this information.
`section` An index number identifying a section in `tableView`.

Returns: The number of rows in section.

Availability: iOS (8.1 and later)

Related Declarations: `tableView(_:titleForHeaderInSection:)`
`tableView(_:cellForRowIndexPath:)`
`tableView`
Show more...

Declared In: UIKit

Reference: [UITableViewDataSource Protocol Reference](#)

UITABLEVIEWDATASOURCE

The screenshot shows an Xcode interface with the following details:

- Toolbar:** Standard Xcode toolbar with icons for file operations.
- Status Bar:** GitHubDemo | Build GitHubDemo: Succeeded | Today at 1:52 AM
- Navigation Bar:** GitHubDemo > GitHubDemo > GitHubMembersTableViewController.swift > tableView(_:cellForRowAtIndexPath:)
- Code View:** Swift code for a UITableViewDataSource implementation:

```
// MARK: - Table view data source

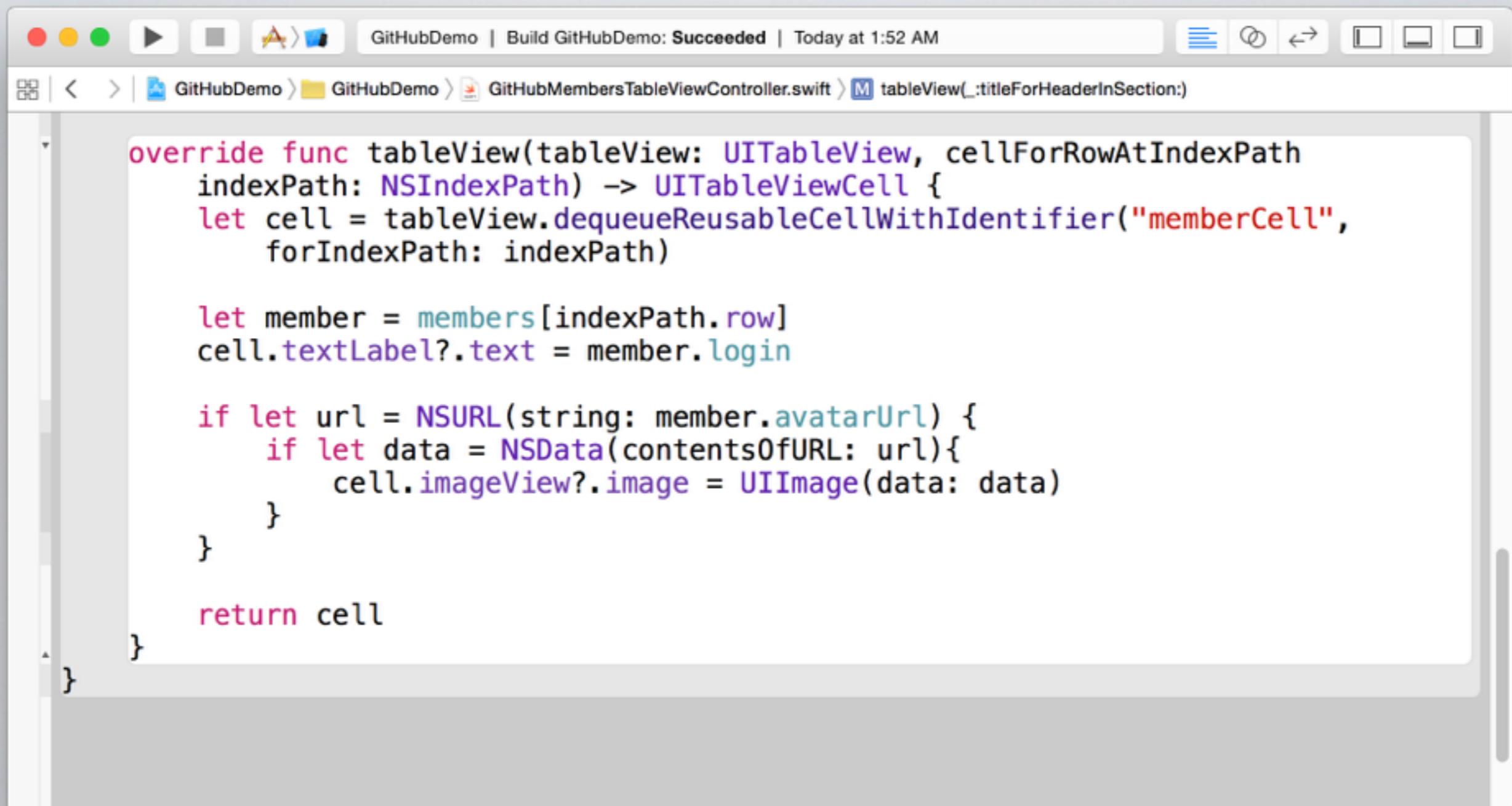
override func numberOfSectionsInTableView(tableView: UITableView) -> Int { ... }

override func tableView(tableView: UITableView, titleForHeaderInSection section: Int) -> String? { ... }

override func tableView(tableView: UITableView, numberOfRowsInSection section: Int) -> Int { ... }

override func tableView(tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell { ... }
```
- Callout:** A callout box is open over the `cellForRowAt indexPath:` method, providing detailed documentation.
 - Declaration:** `override func tableView(tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell`
 - Description:** Asks the data source for a cell to insert in a particular location of the table view.
 - A table-view object requesting the cell.
 - Parameters:**
 - tableView** A table-view object requesting the cell.
 - indexPath** An index path locating a row in `tableView`.
 - Returns:** An object inheriting from `UITableViewCell` that the table view can use for the specified row.
 - Availability:** iOS (8.1 and later)
 - Related Declarations:** `tableView(_:titleForHeaderInSection:)`, `tableView(_:numberOfRowsInSection:)`, `tableView`, `Show more...`
 - Declared In:** UIKit

UITABLEVIEWDATASOURCE



A screenshot of an Xcode interface. The title bar shows "GitHubDemo | Build GitHubDemo: Succeeded | Today at 1:52 AM". The navigation bar below shows the file path "GitHubDemo > GitHubDemo > GitHubMembersTableViewController.swift" and the selected function "tableView(_:titleForHeaderInSection:)". The main editor area contains the following Swift code:

```
override func tableView(tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
    let cell = tableView.dequeueReusableCell(withIdentifier: "memberCell",
    forIndexPath: indexPath)

    let member = members[indexPath.row]
    cell.textLabel?.text = member.login

    if let url = NSURL(string: member.avatarUrl) {
        if let data = NSData(contentsOfURL: url){
            cell.imageView?.image = UIImage(data: data)
        }
    }

    return cell
}
```

UITABLEVIEWDATASOURCE

The screenshot shows the Xcode interface with the title bar "GitHubDemo | Build GitHubDemo: Succeeded | Today at 1:52 AM". The navigation bar indicates the file path "GitHubDemo > GitHubDemo > GitHubMembersTableViewController.swift" and the selected method "tableView(_:cellForRowIndexPath:)". A callout bubble from the Xcode interface highlights the `dequeueReusableCell(withIdentifier: indexPath:)` method. The documentation for this method is displayed in the callout, detailing its declaration, parameters, description, and related declarations.

```
override func tableView(tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
    let cell = tableView.dequeueReusableCell(withIdentifier: "memberCell",
    f Declaration func dequeueReusableCellWithIdentifier(identifier: String, forIndexPath indexPath: IndexPath) ->
    f Description Returns a reusable table-view cell object for the specified reuse identifier.
    let m cell.
    if le i
    }
    return
}
```

Declaration: `func dequeueReusableCellWithIdentifier(identifier: String, forIndexPath indexPath: IndexPath) -> UITableViewCell`

Description: Returns a reusable table-view cell object for the specified reuse identifier.

Parameters:

- identifier**: A string identifying the cell object to be reused. This parameter must not be nil.
- indexPath**: The index path specifying the location of the cell. The data source receives this information when it is asked for the cell and should just pass it along. This method uses the index path to perform additional configuration based on the cell's position in the table view.

Returns: A `UITableViewCell` object with the associated reuse identifier.

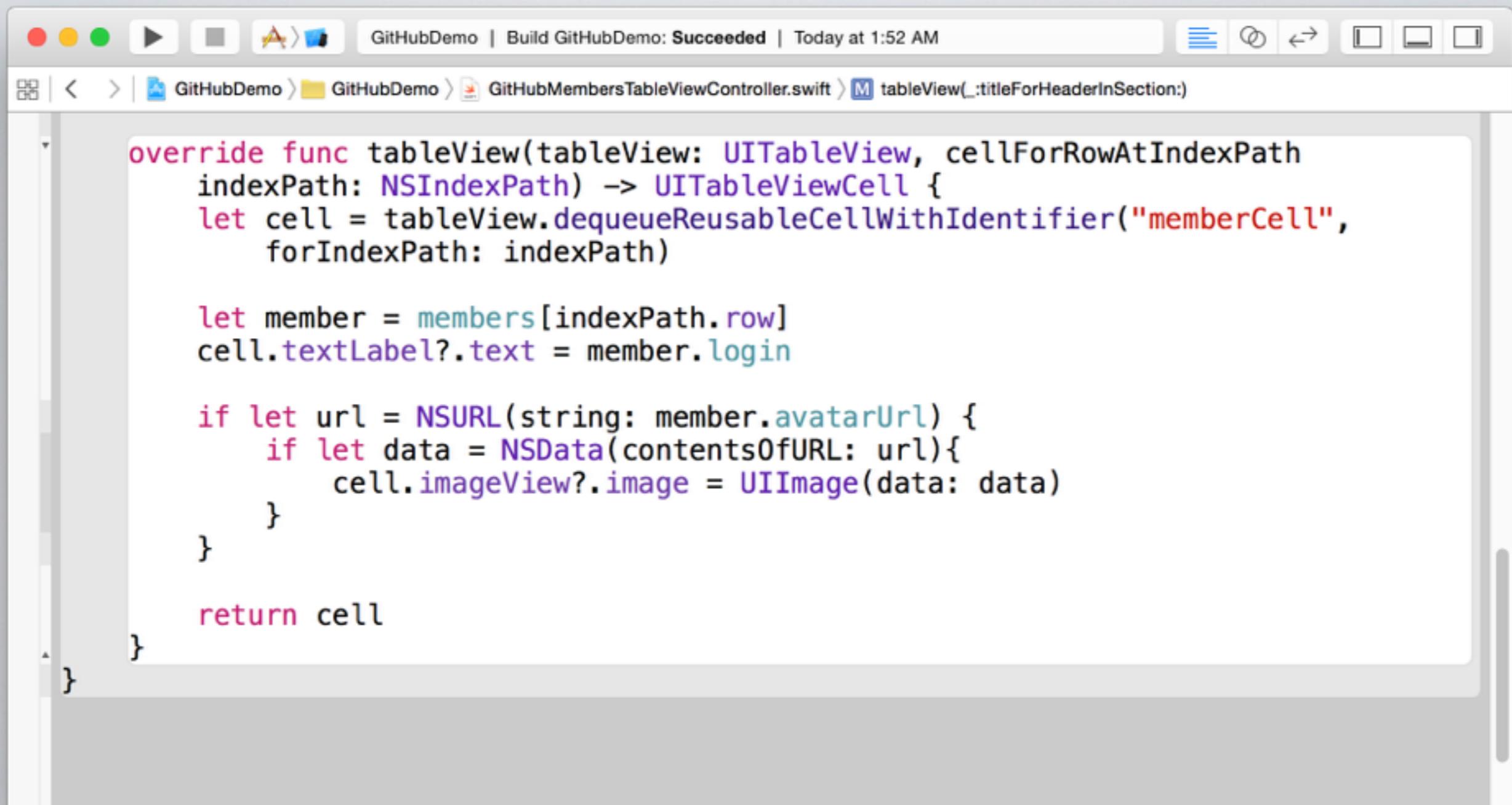
Availability: iOS (6.0 and later)

Related Declarations: `dequeueReusableCell(withIdentifier: _:)`

Declared In: UIKit

Reference: [UITableView Class Reference](#)

UITABLEVIEWDATASOURCE



A screenshot of an Xcode interface. The title bar shows "GitHubDemo | Build GitHubDemo: Succeeded | Today at 1:52 AM". The navigation bar below shows the file path "GitHubDemo > GitHubDemo > GitHubMembersTableViewController.swift" and the selected function "tableView(_:titleForHeaderInSection:)". The main editor area contains the following Swift code:

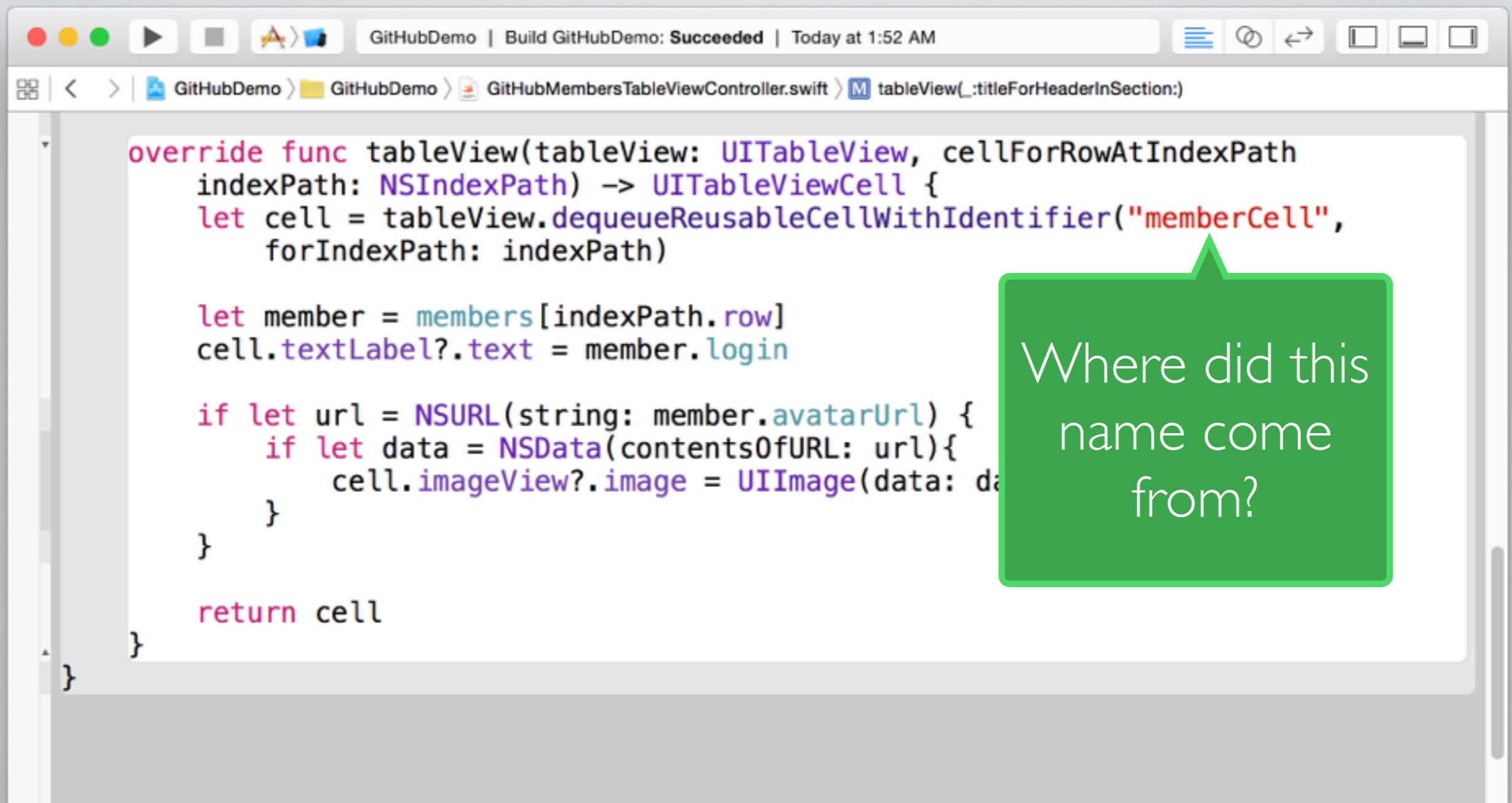
```
override func tableView(tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
    let cell = tableView.dequeueReusableCell(withIdentifier: "memberCell",
    forIndexPath: indexPath)

    let member = members[indexPath.row]
    cell.textLabel?.text = member.login

    if let url = NSURL(string: member.avatarUrl) {
        if let data = NSData(contentsOfURL: url){
            cell.imageView?.image = UIImage(data: data)
        }
    }

    return cell
}
```

UITABLEVIEWDATASOURCE



GitHubDemo | Build GitHubDemo: Succeeded | Today at 1:52 AM

GitHubDemo > GitHubDemo > GitHubMembersTableViewController.swift > tableView(_:titleForHeaderInSection:)

```
override func tableView(tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
    let cell = tableView.dequeueReusableCell(withIdentifier: "memberCell",
    forIndexPath: indexPath)

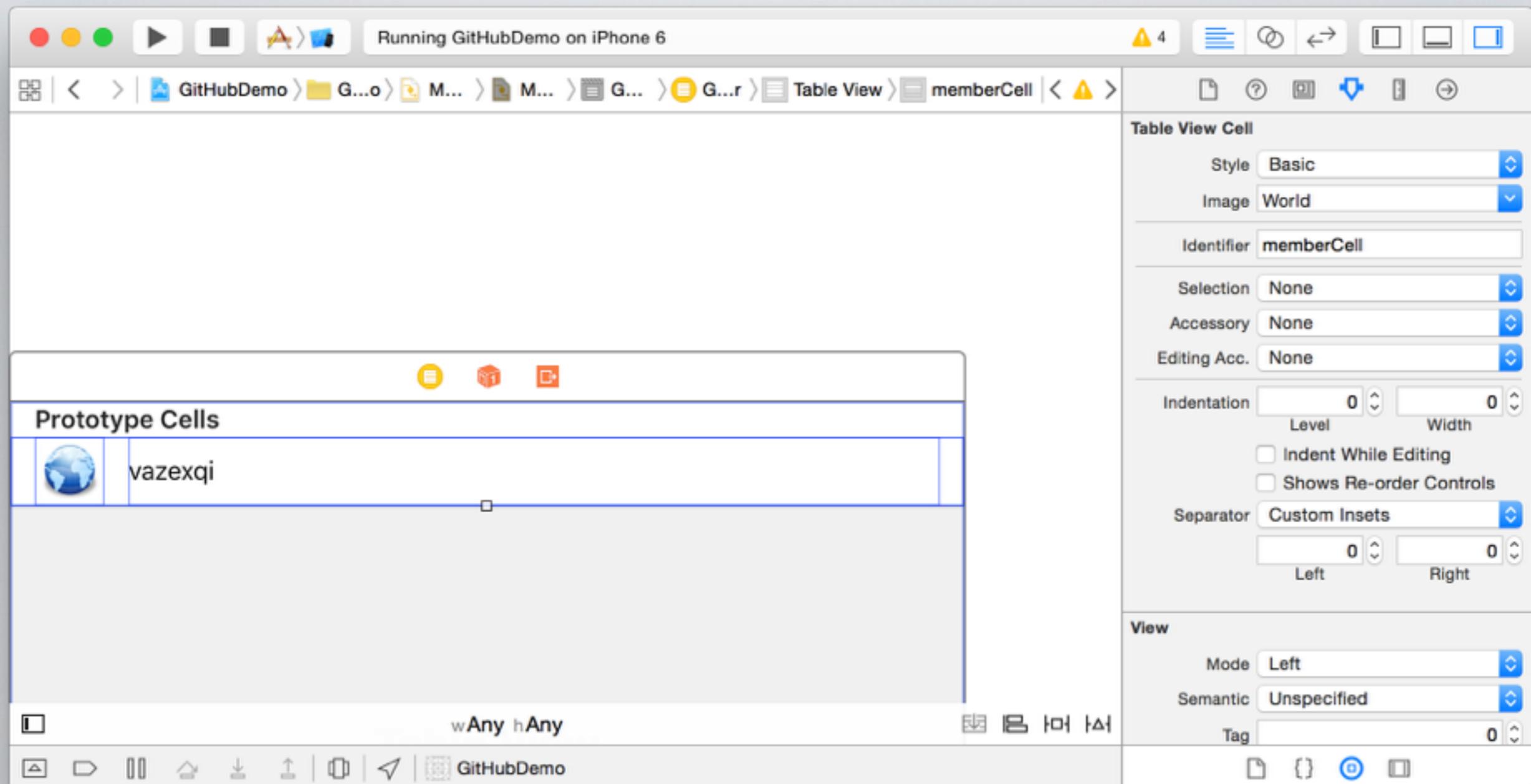
    let member = members[indexPath.row]
    cell.textLabel?.text = member.login

    if let url = NSURL(string: member.avatarUrl) {
        if let data = NSData(contentsOfURL: url){
            cell.imageView?.image = UIImage(data: data)
        }
    }

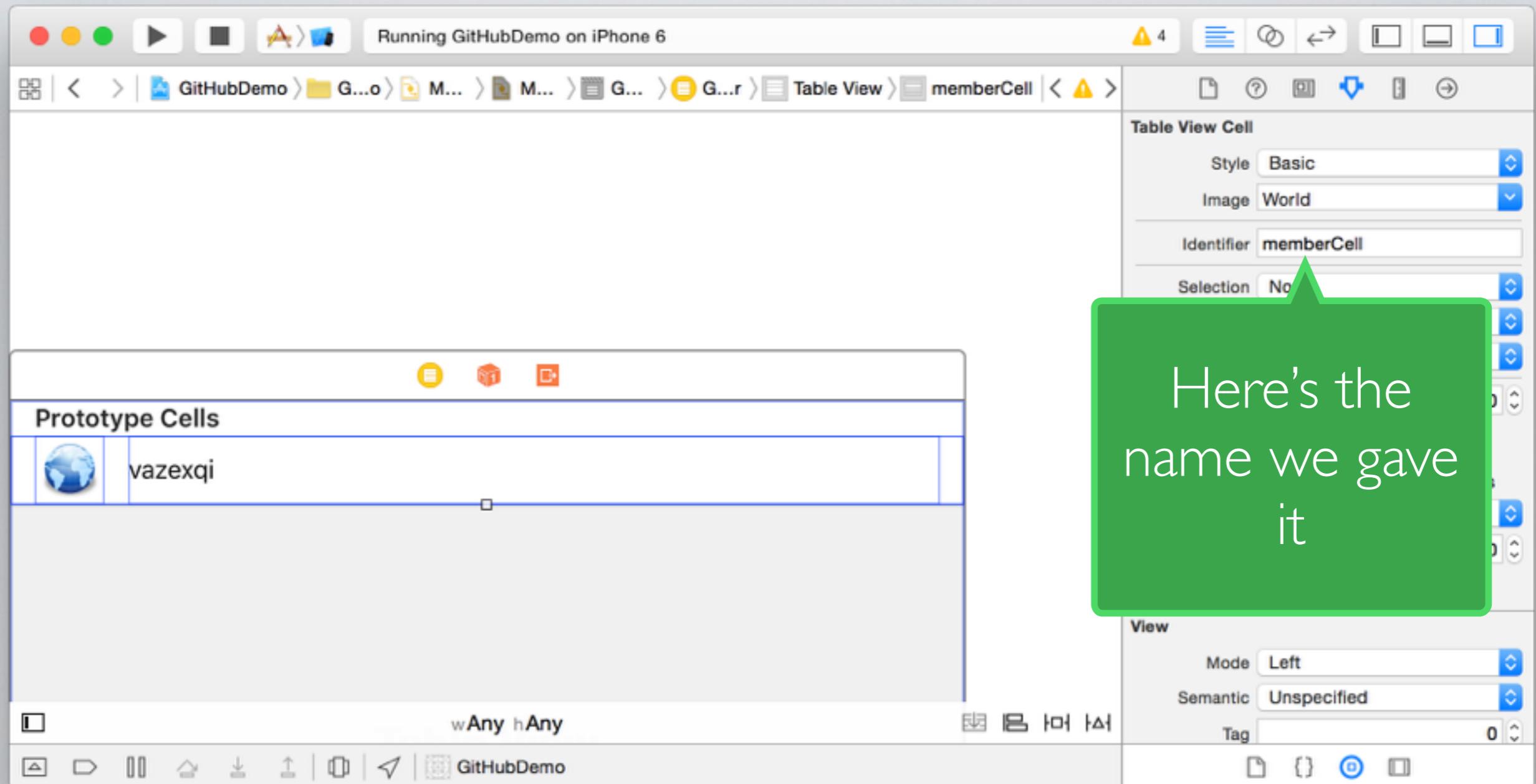
    return cell
}
```

Where did this name come from?

UITABLEVIEWDATASOURCE



UITABLEVIEWDATASOURCE



UITABLEVIEWDELEGATE

UITableViewController

Q uitableviewdelegate

UITableViewController Protocol Reference

Availability

Available in iOS 2.0 and later

The delegate of a UITableView object must adopt the UITableViewDelegate protocol. Optional methods of the protocol allow the delegate to manage selections, configure section headings and footers, help to delete and reorder cells, and perform other actions.

Configuring Rows for the Table View

- tableView:heightForRowAtIndexPath:
- tableView:estimatedHeightForRowAtIndexPath:
- tableView:indentationLevelForRowAtIndexPath:
- tableView:willDisplayCell:forRowAtIndexPath:

Managing Accessory Views

- tableView:editActionsForRowAtIndexPath:
- tableView:accessoryTypeForRowWithIndexPath:
- tableView:accessoryButtonForRowWithIndexPath:

Managing Selections

- tableView:willSelectRowAtIndexPath:



UITABLEVIEWDELEGATE

UITableViewController

uitableviewdelegate

UITableViewController Protocol Reference

Availability

Available in iOS 2.0 and later

The delegate of a UITableView object must adopt the UITableViewDelegate protocol. Optional methods of the protocol allow the delegate to manage selections, configure section headings and footers, help to delete and reorder cells, and perform other actions.

Configuring Rows for the Table View

- tableView:heightForRowAtIndexPath:
- tableView:estimatedHeightForRowAtIndexPath:
- tableView:indentationLevelForRowAtIndexPath:
- tableView:willDisplayCell:forRowAtIndexPath:

Managing Accessory Views

- tableView:editActionsForRowAtIndexPath:
- tableView:accessoryTypeForRowWithIndexPath:
- tableView:accessoryButtonForRowWithIndexPath:

Managing Selections

- tableView:cellForRowAtIndexPath:

Didn't use it this time

CODE

A screenshot of a GitHub repository page for 'talentsparkio / GitHubDemo'. The page shows basic repository statistics: 15 commits, 1 branch, 4 releases, and 1 contributor. A red circle highlights the 'Tag: BasicTableView' dropdown menu. The repository description is 'A quick demo of using the GitHub API to make a simple iOS app — Edit'. The commit list includes:

- vazexqi authored a day ago (latest commit)
- GitHubDemo.xcodeproj Initial working version a day ago
- GitHubDemo.xcworkspace Initial import 10 days ago
- GitHubDemo Initial working version a day ago

CODE

The image shows a split-screen view. On the left is a screenshot of a GitHub repository page for 'talentsparkio / GitHubDemo'. The page displays basic repository statistics: 19 commits, 1 branch, and 4 releases. A dropdown menu is open, showing the current tag 'CircularAvatars' with a red oval highlighting it. Below the dropdown, there's a commit message: 'Make the avatars circular' by user 'vazexqi'. A list of files follows, including 'GitHubDemo.xcodeproj', 'GitHubDemo.xcworkspace', and 'GitHubDemo'. At the bottom, a link points to the commit details. On the right is a screenshot of an iPhone X simulator displaying the GitHub company profile. The profile picture is a black cat silhouette, and the company name is 'GitHub'. It lists 14 members with their names and profile pictures: achiu, adelcambre, aden, aitchabee, alysonla, amateurhuman, ammeep, antonio, arfon, armon, and aroben.

iPhone 6 - iPhone 6 / iOS 9.0 (13A4325c)
Carrier 11:59 AM

This repository Search Pull requests Issues

talentsparkio / GitHubDemo

A quick demo of using the GitHub API to make a simple iOS app — Edit

19 commits 1 branch 4 releases

Tag: CircularAvatars

GitHubDemo / +

Make the avatars circular

vazexqi authored a minute ago

GitHubDemo.xcodeproj Initial working version

GitHubDemo.xcworkspace Initial import

GitHubDemo Make the avatars circular

Open "https://github.com/talentsparkio/GitHubDemo/commits/CircularAvatars" in a new tab

GitHub
GitHub, the company.
San Francisco, CA

Members

- achiu
- adelcambre
- aden
- aitchabee
- alysonla
- amateurhuman
- ammeep
- antonio
- arfon
- armon
- aroben

BONUS (GCD)

The screenshot shows the Xcode IDE running on a Mac. The title bar says "Running GitHubDemo on iPhone 6". The navigation bar shows the project structure: GitHubDemo > GitHubDemo > GitHubMembersTableViewController.swift. The code editor displays the following Swift code:

```
NSIndexPath) -> UITableViewCell {
    let cell = tableView.dequeueReusableCellWithIdentifier("memberCell",
        forIndexPath: indexPath)

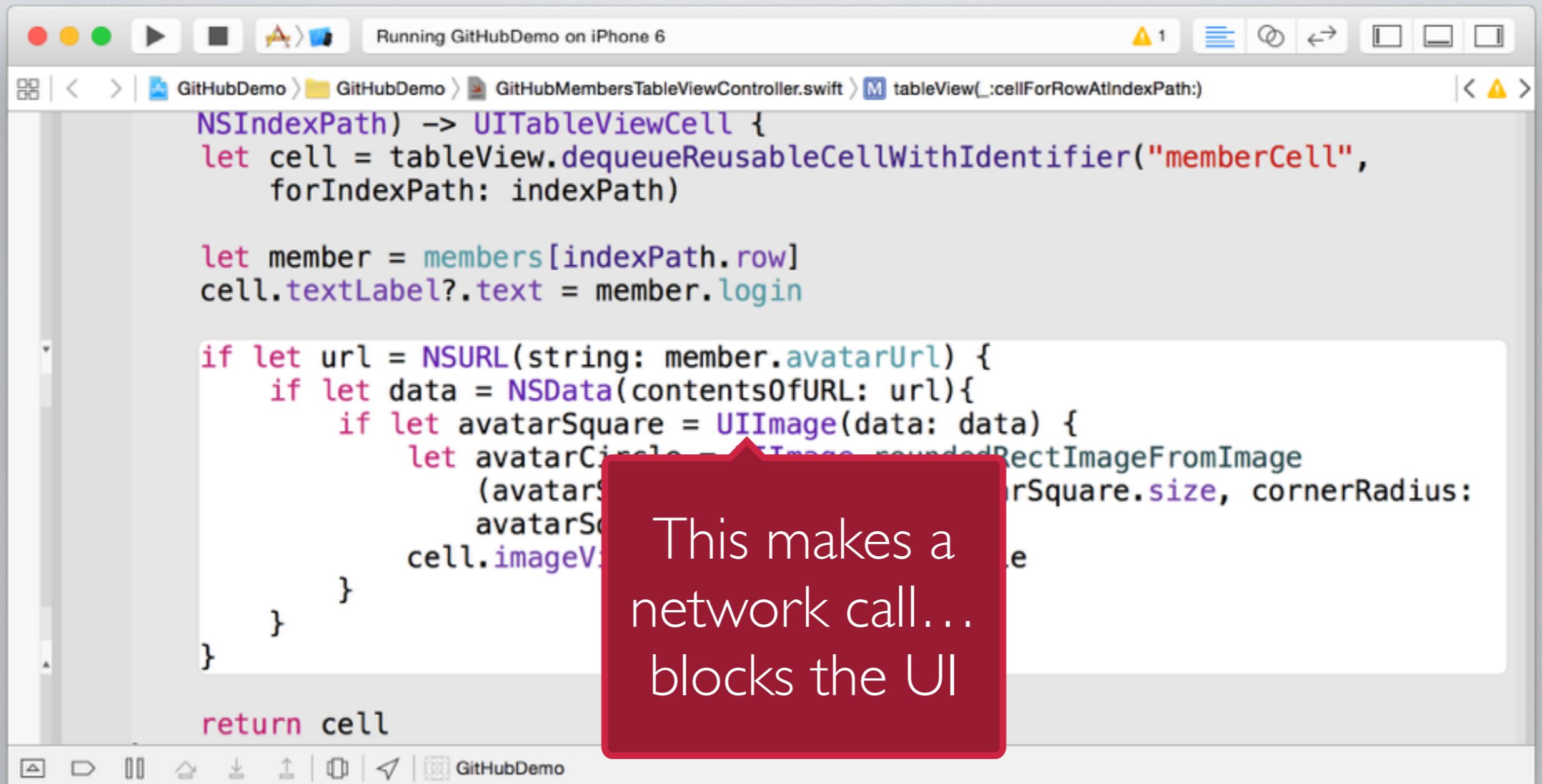
    let member = members[indexPath.row]
    cell.textLabel?.text = member.login

    if let url = NSURL(string: member.avatarUrl) {
        if let data = NSData(contentsOfURL: url){
            if let avatarSquare = UIImage(data: data) {
                let avatarCircle = UIImage.roundedRectImageFromImage
                    (avatarSquare, imageSize: avatarSquare.size, cornerRadius:
                        avatarSquare.size.width / 2)
                cell.imageView?.image = avatarCircle
            }
        }
    }

    return cell
}
```

The code implements a UITableViewDataSource method to dequeue a reusable cell for each row in a UITableView. It retrieves a member from an array of members based on the index path. It then checks if the member has an avatar URL. If so, it creates a square UIImage from the data and rounds its corners to create a circular image. Finally, it sets the text label of the cell to the member's login name and its image view to the circular avatar.

BONUS (GCD)



```
Running GitHubDemo on iPhone 6
GitHubDemo GitHubDemo GitHubMembersTableViewController.swift tableView(_:cellForRowAtIndexPath:)

NSIndexPath) -> UITableViewCell {
    let cell = tableView.dequeueReusableCellWithIdentifier("memberCell",
        forIndexPath: indexPath)

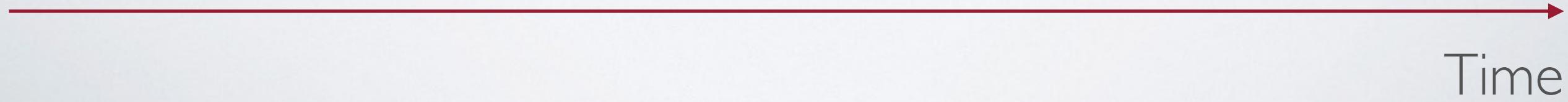
    let member = members[indexPath.row]
    cell.textLabel?.text = member.login

    if let url = NSURL(string: member.avatarUrl) {
        if let data = NSData(contentsOfURL: url){
            if let avatarSquare = UIImage(data: data) {
                let avatarCircle = UIImage.roundedRectImageFromImage
                    (avatarSquare, cornerRadius: avatarSquare.size, cornerRadius:
                avatarSquare)
                cell.imageView.image = avatarCircle
            }
        }
    }
    return cell
}
```

This makes a
network call...
blocks the UI

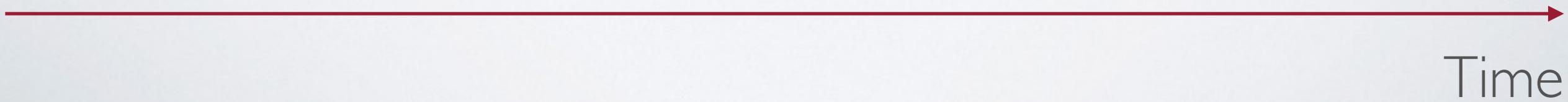
BONUS (GCD)

BONUS (GCD)



BONUS (GCD)

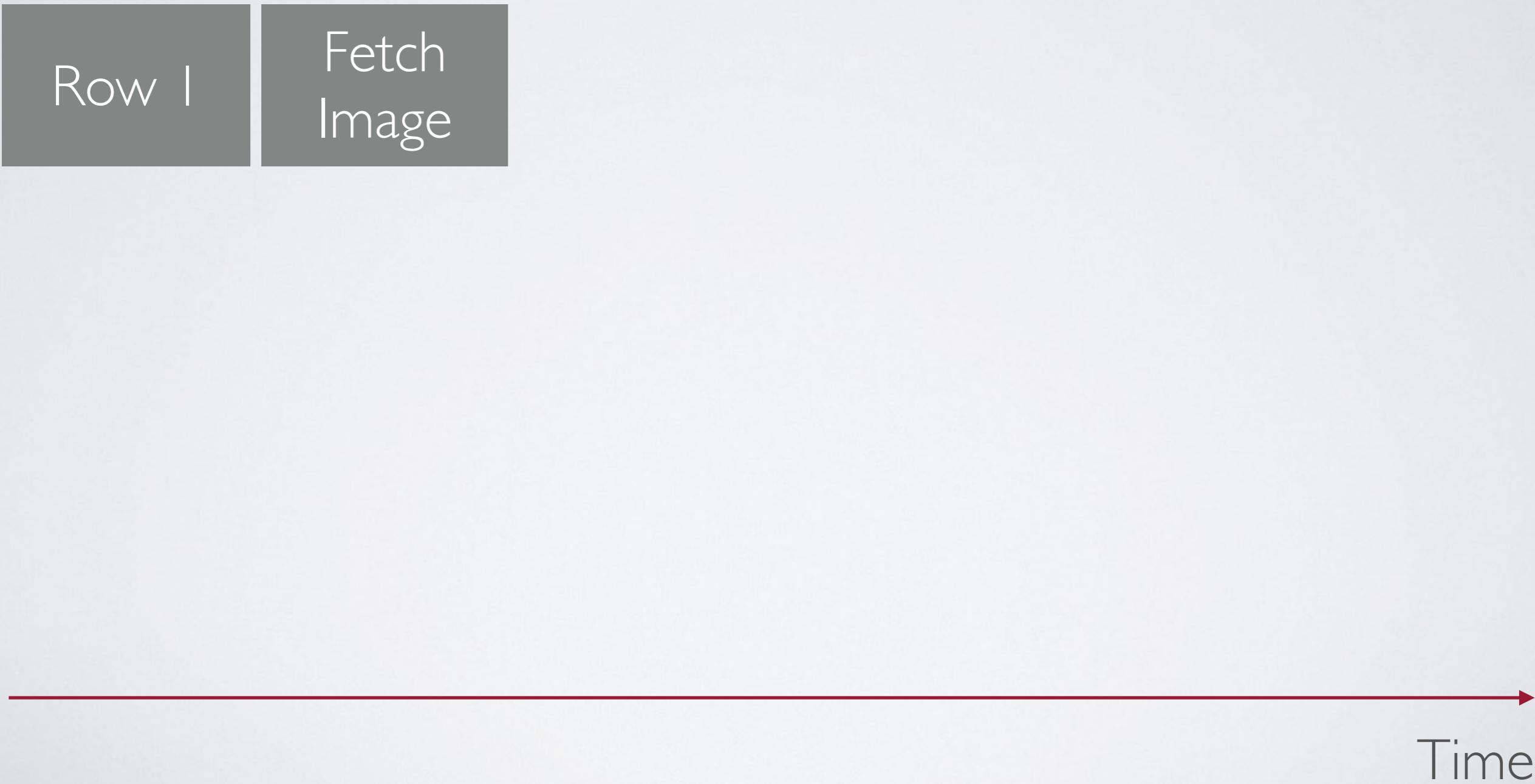
Row I



BONUS (GCD)

Row I

Fetch
Image



BONUS (GCD)



Time

BONUS (GCD)



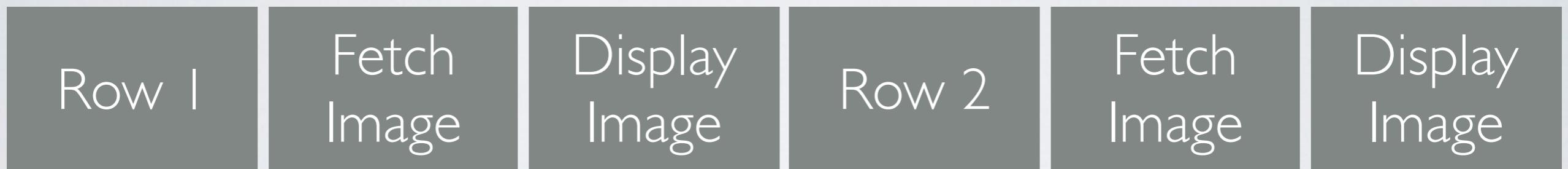
Time

BONUS (GCD)

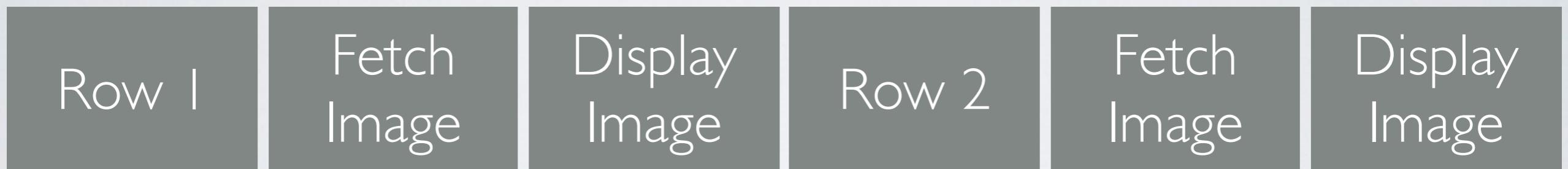


Time

BONUS (GCD)



BONUS (GCD)



Can't display
Row 2 until
Row 1 finishes

Time

BONUS (GCD)



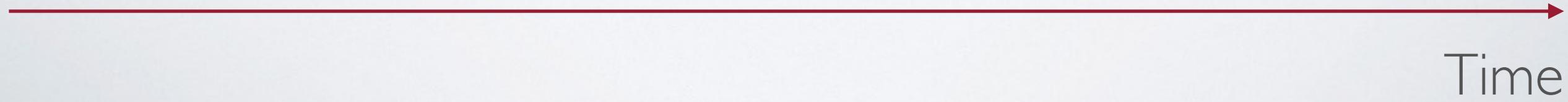
Can't display
Row 2 until
Row 1 finishes



Can we at least display the text first and images later?

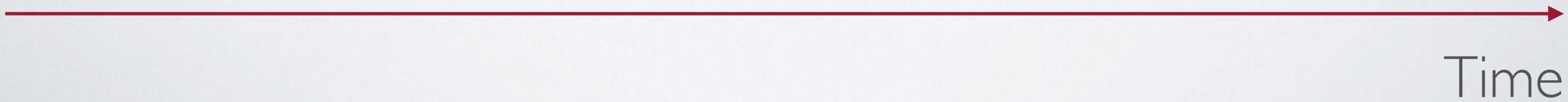
Time

BONUS (GCD)



BONUS (GCD)

Row I



BONUS (GCD)

Row 1

Row 2

Time

BONUS (GCD)

Row 1

Row 2

Row 3

Time

BONUS (GCD)

Row 1

Row 2

Row 3

...

Time

BONUS (GCD)

Row 1

Row 2

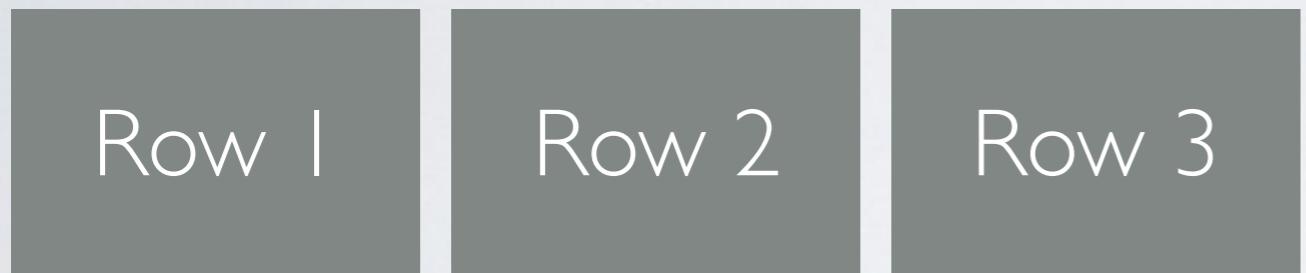
Row 3

...

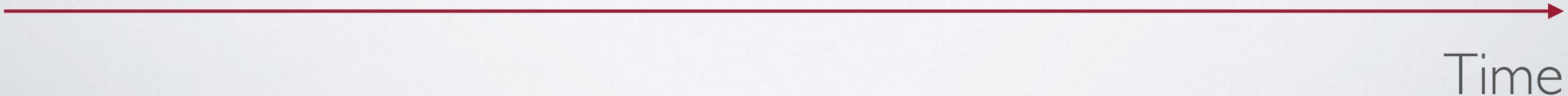
Some time later

Time

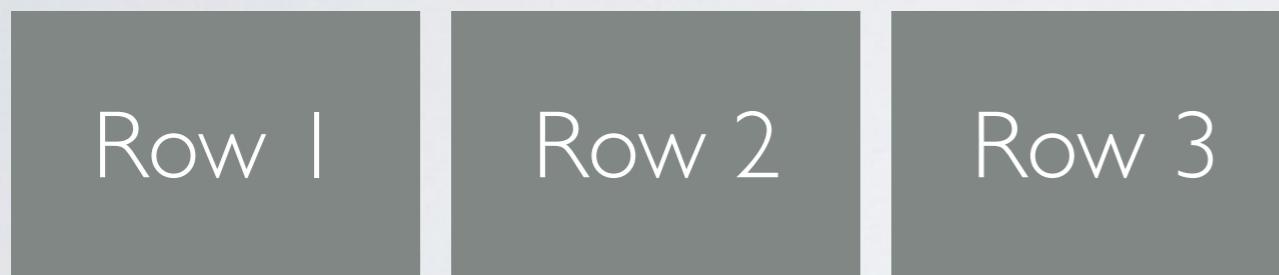
BONUS (GCD)



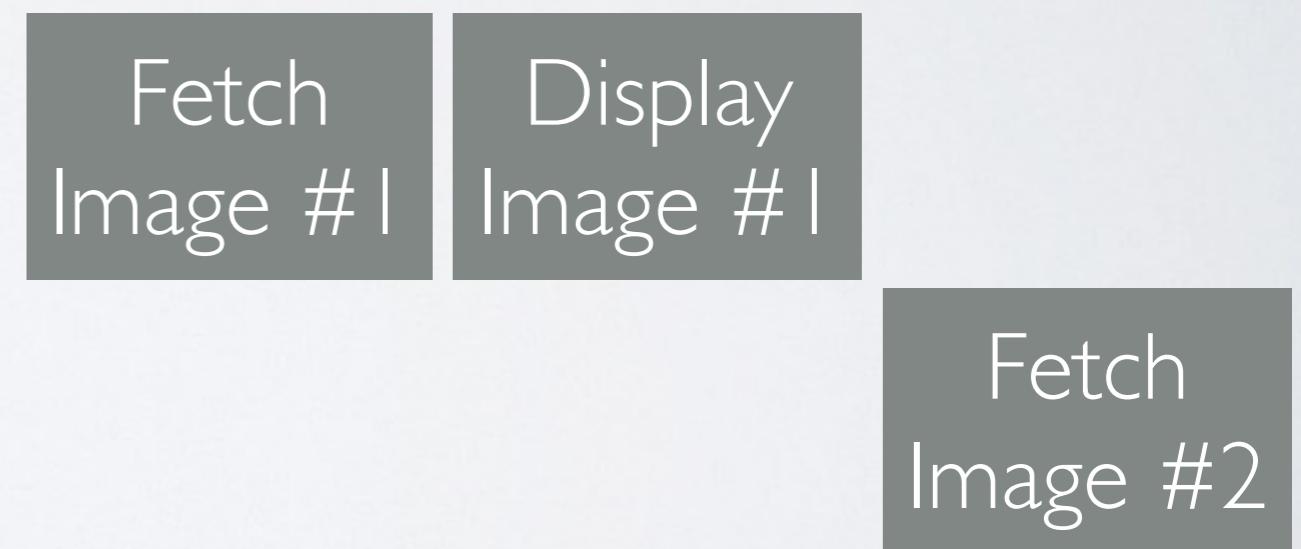
Some time later



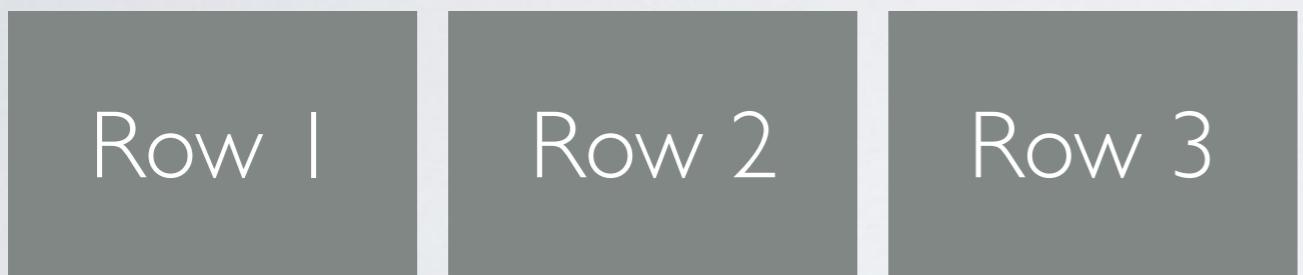
BONUS (GCD)



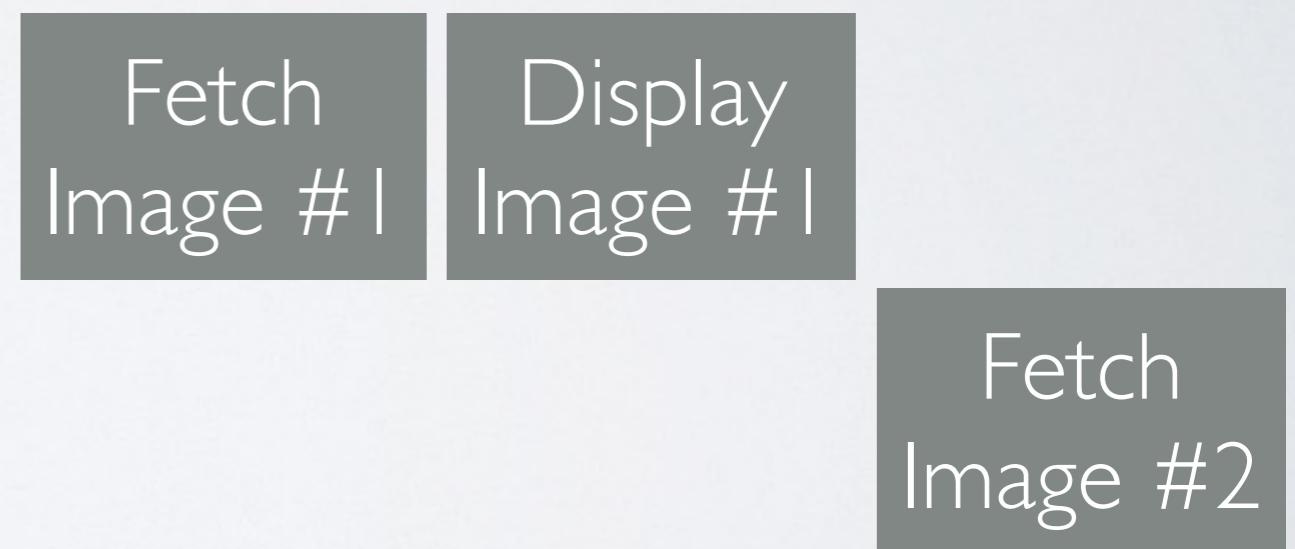
Some time later



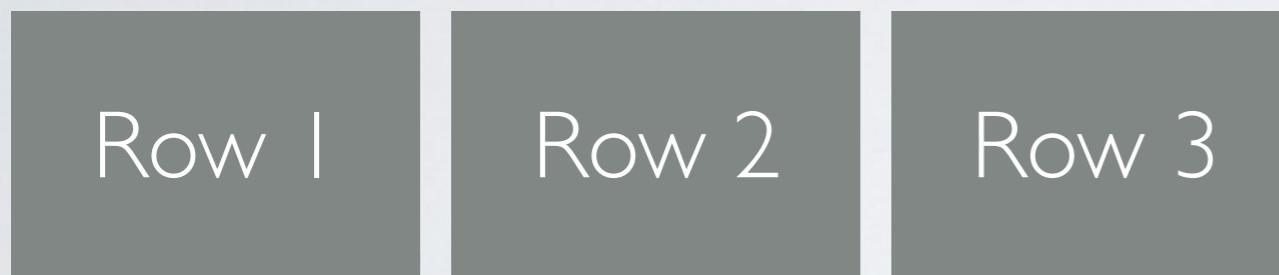
BONUS (GCD)



Some time later



BONUS (GCD)

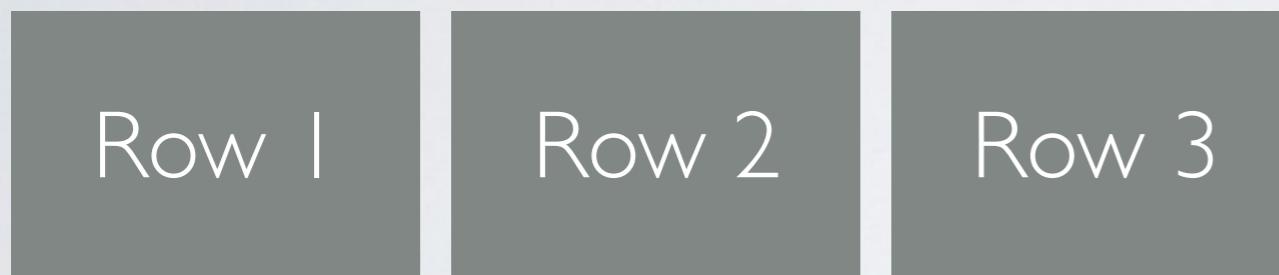


Some time later

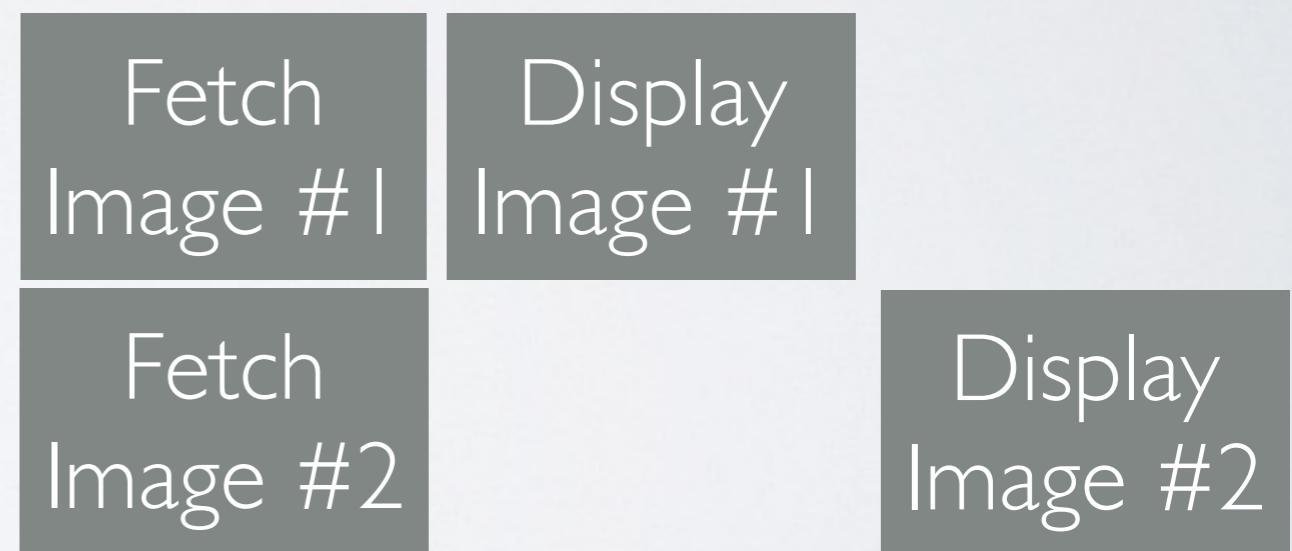


Time

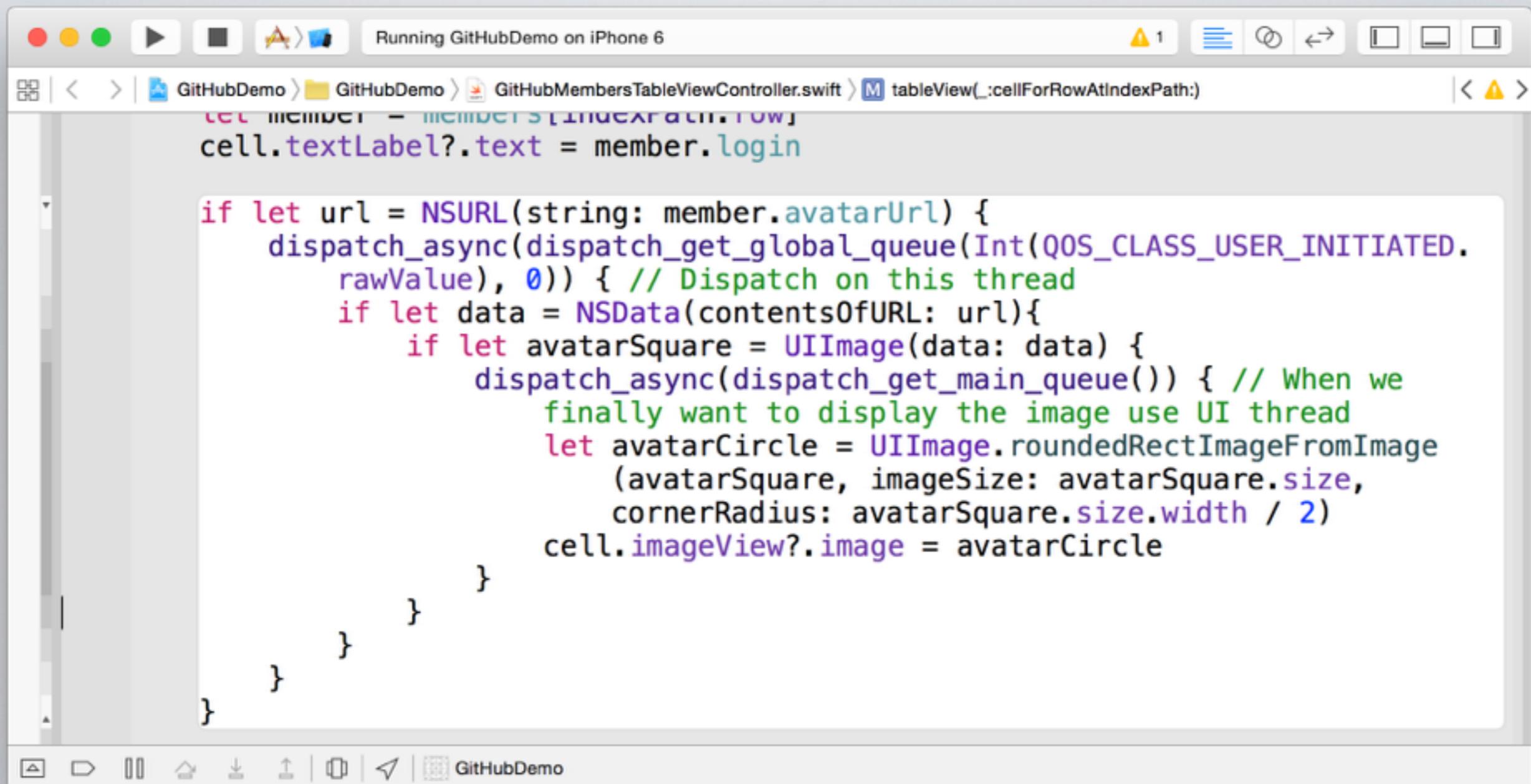
BONUS (GCD)



Some time later



BONUS (GCD)



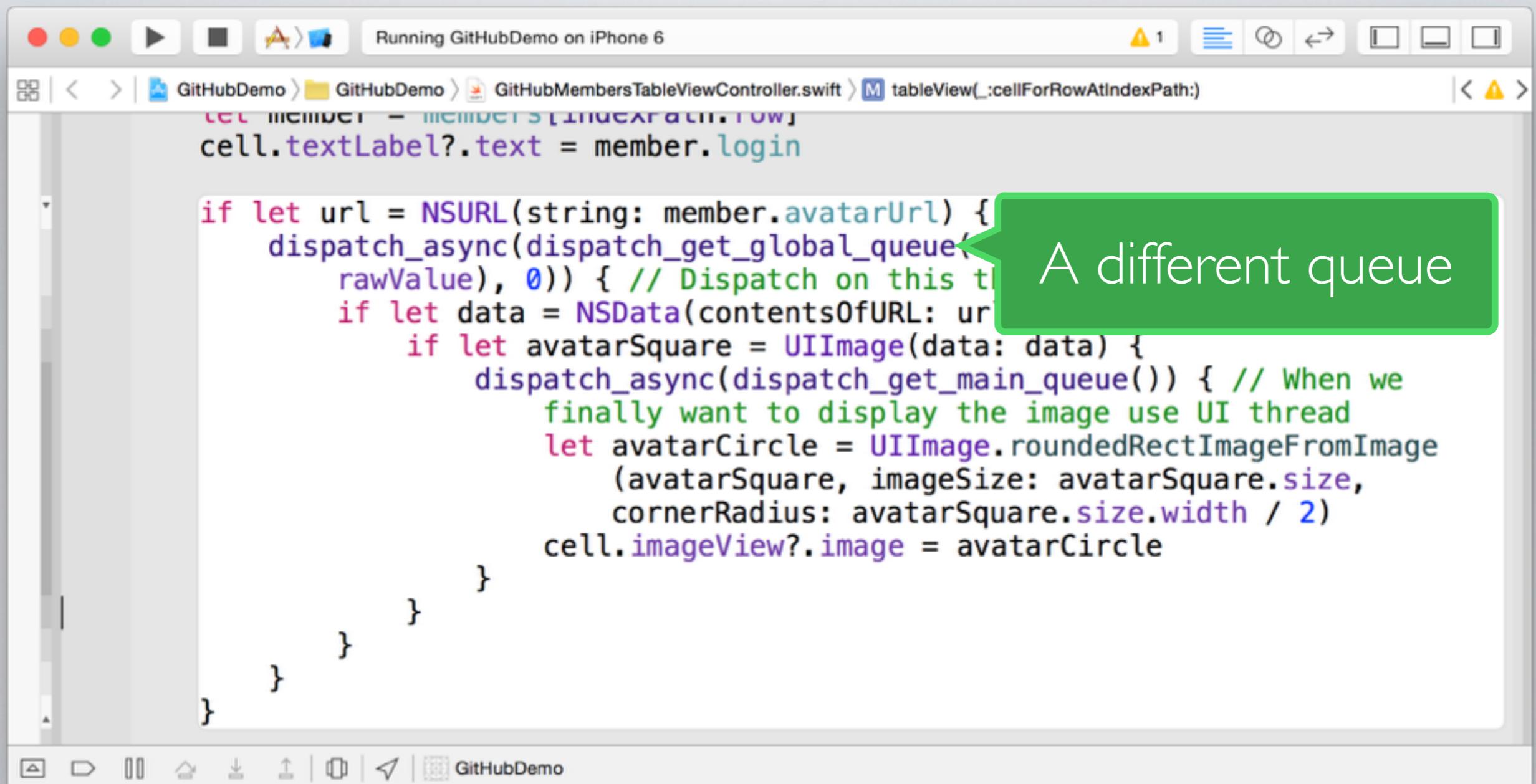
The screenshot shows the Xcode IDE running on a Mac. The title bar says "Running GitHubDemo on iPhone 6". The navigation bar shows the project structure: GitHubDemo > GitHubDemo > GitHubMembersTableViewController.swift. The code editor displays the following Swift code:

```
func tableView(_: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
    let member = members[indexPath.row]
    let cell = tableView.dequeueReusableCell(withIdentifier: "MemberCell", for: indexPath)
    cell.textLabel?.text = member.login

    if let url = NSURL(string: member.avatarUrl) {
        dispatch_async(dispatch_get_global_queue(Int(QOS_CLASS_USER_INITIATED.rawValue), 0)) { // Dispatch on this thread
            if let data = NSData(contentsOfURL: url){
                if let avatarSquare = UIImage(data: data) {
                    dispatch_async(dispatch_get_main_queue()) { // When we
                        finally want to display the image use UI thread
                        let avatarCircle = UIImage.roundedRectImageFromImage(
                            avatarSquare, imageSize: avatarSquare.size,
                            cornerRadius: avatarSquare.size.width / 2)
                    cell.imageView?.image = avatarCircle
                }
            }
        }
    }
}
```

The code implements a UITableViewDataSource method to configure a table view cell. It retrieves a member from an array and sets the cell's text label. Then it checks if the member has an avatar URL. If so, it uses GCD to download the image from the URL. Once the image is downloaded, it creates a rounded circular image and sets it as the cell's image view. The code uses `dispatch_get_global_queue` for the download and `dispatch_get_main_queue` for displaying the image.

BONUS (GCD)



A screenshot of an Xcode workspace titled "Running GitHubDemo on iPhone 6". The file open is "GitHubMembersTableViewController.swift" at the path "GitHubDemo > GitHubDemo > GitHubMembersTableViewController.swift". The code is as follows:

```
func tableView(_: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
    let member = members[indexPath.row]
    let cell = tableView.dequeueReusableCell(withIdentifier: "MemberCell", for: indexPath)
    cell.textLabel?.text = member.login

    if let url = NSURL(string: member.avatarUrl) {
        dispatch_async(dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT, 0)) { // Dispatch on this thread
            if let data = NSData(contentsOf: url) {
                if let avatarSquare = UIImage(data: data) {
                    dispatch_async(dispatch_get_main_queue()) { // When we finally want to display the image use UI thread
                        let avatarCircle = UIImage.roundedRectImageFromImage(avatarSquare, imageSize: avatarSquare.size, cornerRadius: avatarSquare.size.width / 2)
                        cell.imageView?.image = avatarCircle
                    }
                }
            }
        }
    }
}
```

A green callout bubble with the text "A different queue" points to the line `dispatch_async(dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT, 0))`.

BONUS (GCD)

The screenshot shows an Xcode interface with the title bar "Running GitHubDemo on iPhone 6". The navigation bar indicates the file is "GitHubMembersTableViewController.swift" at "tableView(_:cellForRowAtIndexPath:)". The code is as follows:

```
    cell.textLabel?.text = member.login

    if let url = NSURL(string: member.avatarUrl) {
        dispatch_async(dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT, 0)) { // Dispatch on this thread
            if let data = NSData(contentsOfURL: url) {
                if let avatarSquare = UIImage(data: data)
                    dispatch_async(dispatch_get_main_queue()) { // Finally want to display the image on the main thread
                        let avatarCircle = UIImage.roundedSquareImage(avatarSquare, imageSize: avatarSquare.size,
                            cornerRadius: avatarSquare.size.width / 2)
                        cell.imageView?.image = avatarCircle
                    }
            }
        }
    }
}
```

Two green callout boxes with arrows point to specific lines of code:

- An arrow points to the line `dispatch_async(dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT, 0))` with the text "A different queue".
- An arrow points to the line `dispatch_async(dispatch_get_main_queue())` with the text "The main queue".

NEXT TIME

- Revisit auto-layouts to see some other examples
- How to persist data/edit data in table views