PROTOCOL ORIENTED NETWORKING

WITH

RestoFire

By: Rahul Katariya

@RAHULKATARIYA

- Restofire
- JetpackSwift/FrameworkTemplate
- JetpackSwift/Fastlane

OVERVIEW

CONFIGURING RESTOFIRE

```swift
import UIKit
import Restofire

@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate {

    var window: UIWindow?

    func application(
        _ application: UIApplication,
        didFinishLaunchingWithOptions launchOptions: [UIApplicationLaunchOptionsKey: Any]?
        ) -> Bool
    {
        // Override point for customization after application launch.

        Restofire.defaultConfiguration.baseURL = "http://www.mocky.io/v2/"

        return true
    }

}
```

CREATING A REQUEST

```swift
import Restofire

struct PersonGETService: Requestable {

    typealias Model = [String: Any]
    var path: String = "56c2cc70120000c12673f1b5"

}
```

MAKING A REQUEST

```
import UIKit
import Restofire

class ViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()
        PersonGETService().executeTask() {
            if let value = $0.result.value {
                print(value)
            }
        }
    }
}
```

REQUESTABLE

- Requestable represents an HTTP Request that can be asynchronously executed
- Associates response type to ensure type safety
- Provides default implementations

```swift
public protocol Requestable: Authenticable, Configurable, ResponseSerializable, Retryable, SessionManagable,
Validatable, Queueable {

    /// The response type.
    associatedtype Model
    /// The base URL.
    var baseURL: String { get }
    /// The path relative to base URL.
    var path: String { get }
    /// The HTTP Method.
    var method: Alamofire.HTTPMethod { get }
    /// The request parameter encoding.
    var encoding: Alamofire.ParameterEncoding { get }
    /// The HTTP headers.
    var headers: [String : String]? { get }
    /// The request parameters.
    var parameters: Any? { get }
    /// The Alamofire Session Manager.
    var sessionManager: Alamofire.SessionManager { get }
    /// The queue on which reponse will be delivered.
    var queue: DispatchQueue? { get }
    /// The credential.
    var credential: URLCredential? { get }
    /// The Alamofire validation.
    var validationBlock: Alamofire.DataRequest.Validation? { get }
    /// The acceptable status codes.
    var acceptableStatusCodes: [Int]? { get }
    /// The acceptable content types.
    var acceptableContentTypes: [String]? { get }
    /// The retry error codes.
    var retryErrorCodes: Set<URLError.Code> { get }
    /// The retry interval.
    var retryInterval: TimeInterval { get }
    /// The max retry attempts.
    var maxRetryAttempts: Int { get }
}
```

FEATURES

- Three levels of configurations
  - Global
  - Group
  - Request
- Request Eventually with Auto Retry
- Isolating Requests from UIViewController
- Custom Response Serializer

GLOBAL CONFIGURATION

```
Restofire.defaultConfiguration.baseURL = “http://www.mocky.io/v2/"
Restofire.defaultConfiguration.headers = ["Content-Type": “application/json”]
Restofire.defaultConfiguration.logging = true

Restofire.defaultConfiguration.authentication.credential = URLCredential(user: "user",
password: "password", persistence: .forSession)

Restofire.defaultConfiguration.validation.acceptableStatusCodes = Array(200..<300)
Restofire.defaultConfiguration.validation.acceptableContentTypes = [“application/json"]

Restofire.defaultConfiguration.retry.retryErrorCodes = [.timedOut,.networkConnectionLost]
Restofire.defaultConfiguration.retry.retryInterval = 20
Restofire.defaultConfiguration.retry.maxRetryAttempts = 10

let sessionConfiguration = URLSessionConfiguration.default
sessionConfiguration.timeoutIntervalForRequest = 7
sessionConfiguration.timeoutIntervalForResource = 7
sessionConfiguration.httpAdditionalHeaders = Alamofire.SessionManager.defaultHTTPHeaders
Restofire.defaultConfiguration.sessionManager = Alamofire.SessionManager(configuration:
sessionConfiguration)
```

GROUP CONFIGURATION

```swift
public protocol Requestable: Authenticable, Configurable, ResponseSerializable, Retryable,
SessionManagable, Validatable, Queueable {}


protocol HTTPBinConfigurable: Configurable { }

extension HTTPBinConfigurable {

    var configuration: Configuration {
        var config = Configuration()
        config.baseURL = "https://httpbin.org/"
        return config
    }

}


struct HTTPBinStringGETService: Requestable, HTTPBinConfigurable { }
```

REQUEST CONFIGURATION

```swift
import Restofire
import Alamofire

struct MoviesReviewGETService: Requestable {

    typealias Model = [MovieReview]
    var path: String = "reviews/"
    var parameters: Any?
    var encoding: ParameterEncoding = URLEncoding.default

    init(id: String, parameters: Any) {
        self.path += id
        self.parameters = parameters
    }

}
```

MAKING A REQUEST

```swift
import UIKit
import Restofire

class ViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()
        MoviesReviewGETService(id: “all.json", parameters: ["api-key":key]).executeTask() {
            if let value = $0.result.value {
                print(value)
            }
        }
    }
}
```

CANCELLING A REQUEST

```swift
import UIKit
import Restofire

class ViewController: UIViewController {

     var request: DataRequestOperation<PersonGETService>!

    override func viewDidLoad() {
        super.viewDidLoad()
        request = PersonGETService().executeTask() {
            if let value = $0.result.value {
                print(value)
            }
        }
    }

     deinit {
      request.cancel()
    }

}
```

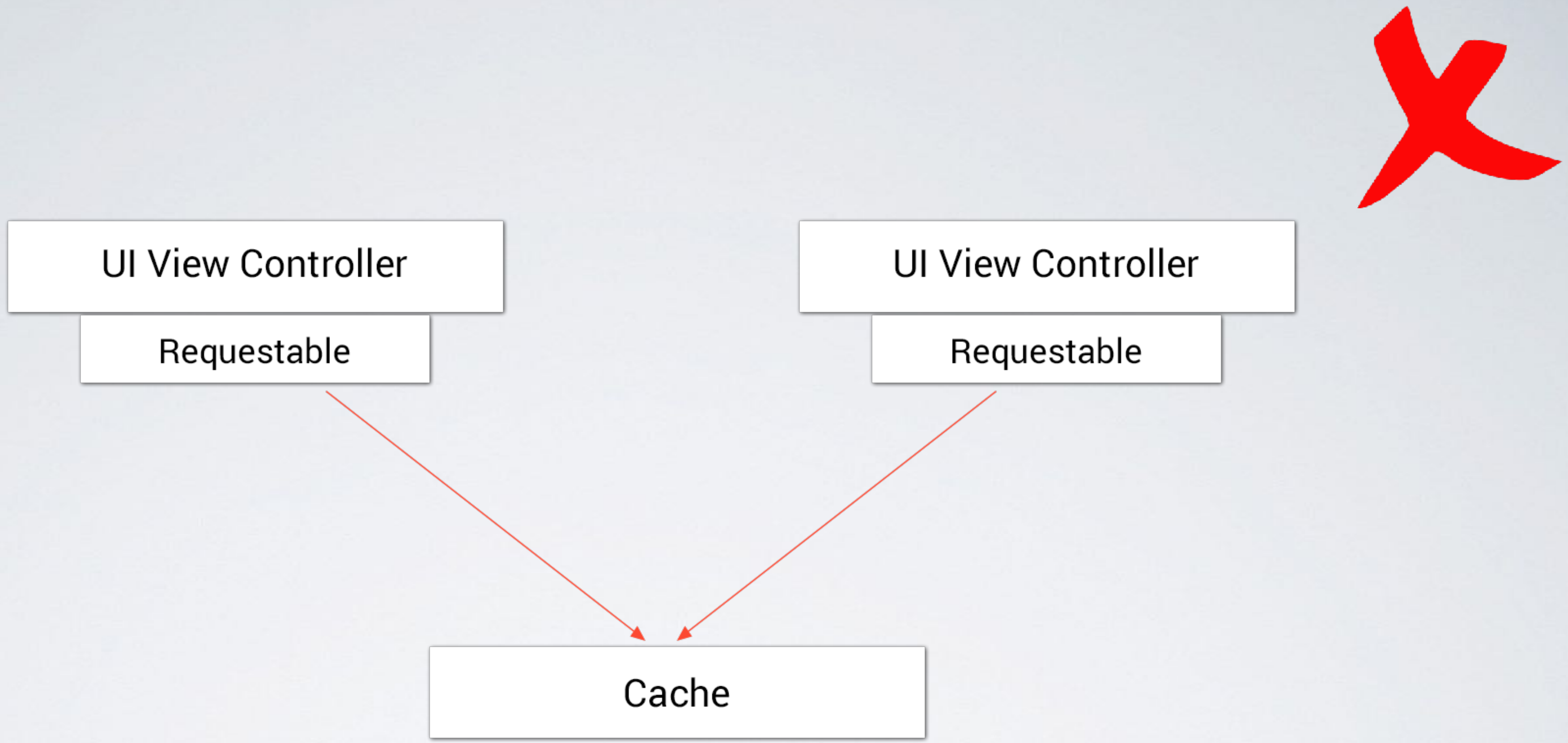
REQUEST EVENTUALLY

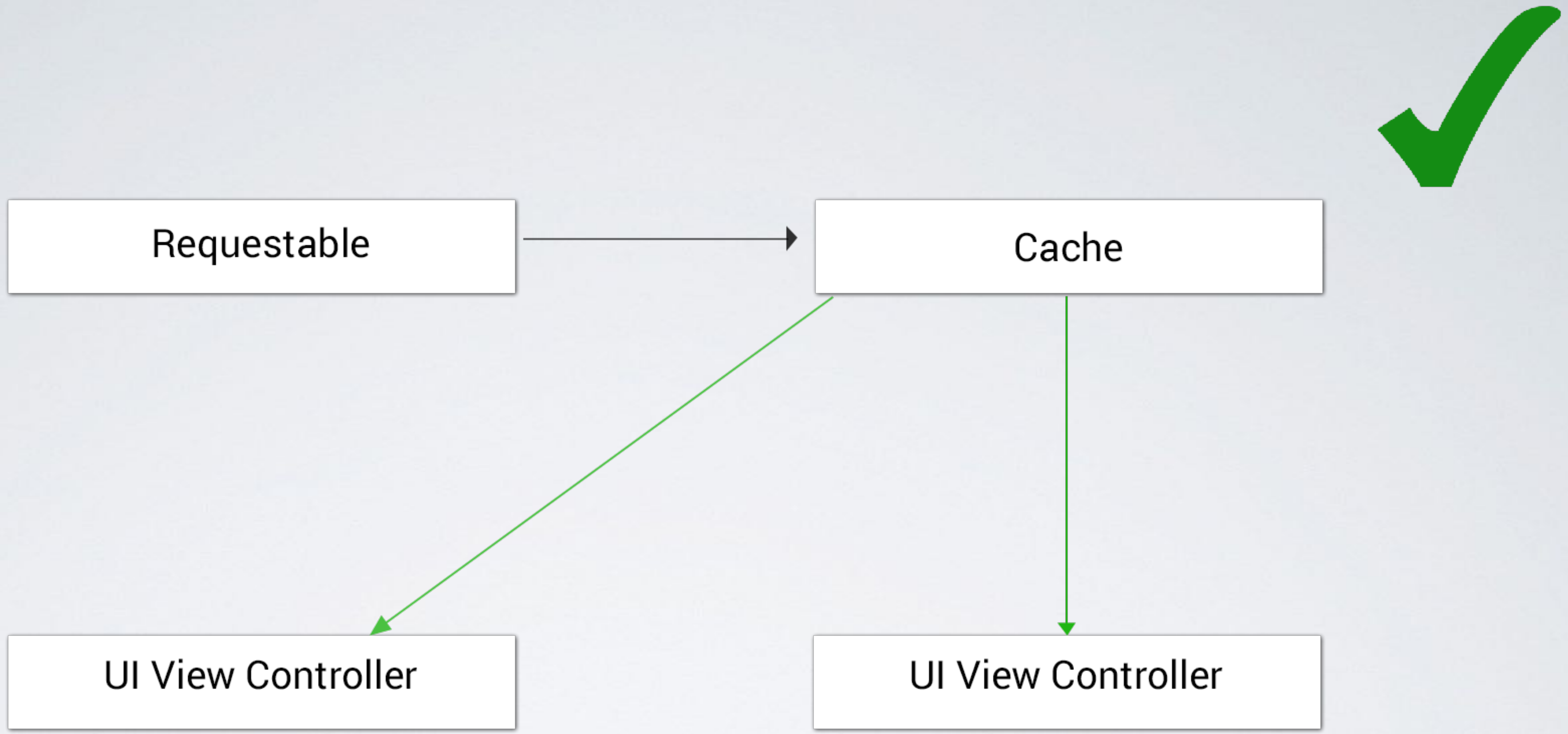
```swift
import UIKit
import Restofire

class ViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()
        PersonGETService().executeTaskEventually() {
            if let value = $0.result.value {
                print(value)
            }
        }
    }
}
```

ISOLATING NETWORK CALLS FROM VIEW CONTROLLER


UI View Controller
Requestable
UI View Controller
Requestable
Cache


Requestable
Cache
UI View Controller
UI View Controller

```swift
// MARK: - Caching
import RealmSwift

extension MoviesReviewGETService {

    func didCompleteRequestWithDataResponse(_ response: DataResponse<Model>) {
        guard let movieReviews = response.result.value else { return }
        let realm = try! Realm()
        for movieReview in movieReviews {
            try! realm.write {
                realm.add(movieReview, update: true)
            }
        }
    }

}
```

QUESTIONS?