“Fake It Til They Make It” - Mocking REST APIs in Swift

Chris Woodard
Suncoast iOS Meetup

REST APIs

REST API environment

REST APIs

- Use HTTP verbs
  - GET, POST, PUT, DELETE
- Use HTTP status codes to describe results (200, 401, 404, 500, etc)
- Use URI nouns and modifiers to denote resources
  - /index/list.php?page=10&size=50
- Consumer facing structure

REST Client

- Knows about the nouns, understands the structure
- Knows which nouns respond to which verbs
- Connects to the web server
- Builds HTTP requests
  - GET /index/list.php?page=10&size=50
- Parses the response and interprets the status code

REST Client in iOS

- Uses the URLSession API & URLRequests
  - URLSession & iOS handle the plumbing
  - Operates nicely with operation queues for background threading
- Supplies the nouns and verbs
- Parses the response and interprets the status code

Sample

```
        let encoded = self.encodedUrlString(str: cityAndCountry)
        if let urlString = URL(string: "https://api.openweathermap.org/data/2.5/forecast?q=\
(encoded)&units=imperial&appid=3a5a5533643dadd75a8c095541dea0ed") {
            var request = URLRequest(url: urlString)
            request.httpMethod = "GET"
            var resultsDict:[String:Any] = [:]
            let task = self.session?.dataTask(with: request, completionHandler: {data, response,
error in
                if let rsp = response as? HTTPURLResponse {
                    resultsDict["Status"] = rsp.statusCode
                    if 200 == rsp.statusCode {
                        if let forecastData = data {
                            if let forecastDict = try? JSONSerialization.jsonObject(with:
forecastData, options: .mutableContainers) as! [String : Any] {
                                resultsDict["Data"] = forecastDict
                            }
                        }
                        else {
                            resultsDict["Status"] = 500
                        }
                    }
                    else {

                    }
                }
                completion(resultsDict, error)
            })
            task?.resume()
        }
```

Demo

So Far So Good, But…

- Manual Testing Only
- Automated Testing *Can* Work, But…
  - Not consistent since forecast changes
  - What if server is offline?
- What if your server team is running behind?
  - Nothing to code against.

What Can We Do?

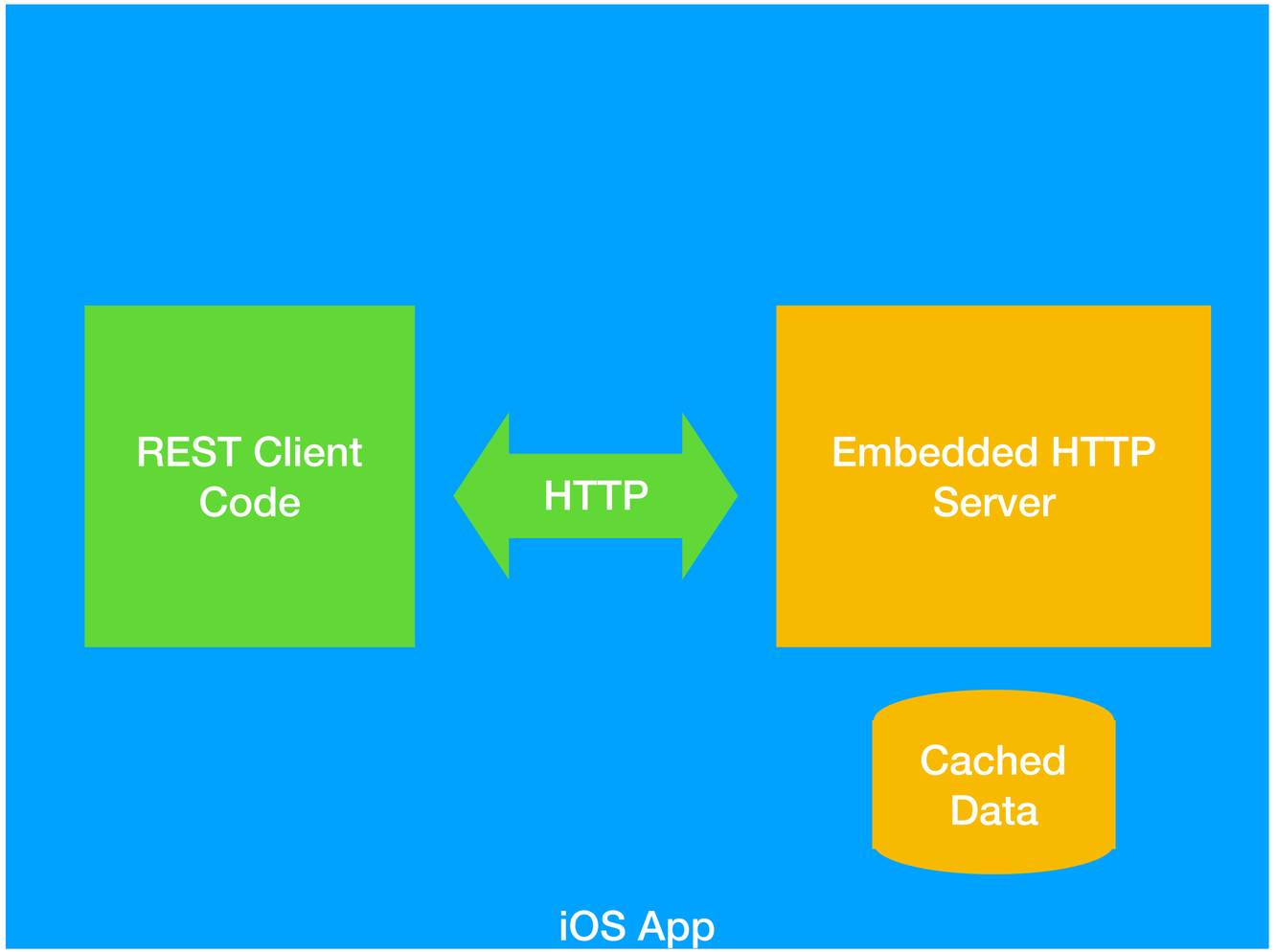
Mock Data, Duh!

- Static data included (or generated) in app
- Loaded and processed instead of live data
- Consistent, always available, but…
- You have to get it somewhere
- You have to store it locally
- You have to tell the app to use it instead of trying to get live data

Method #1 - In The Code

```
-(void)forecastForCity:(NSString *)cityAndCountry completion:(void(^)(NSDictionary *forecast,
NSError *err))completion {

    NSMutableCharacterSet *allowed = [NSMutableCharacterSet
                                      alphanumericCharacterSet];
    NSString *encoded = [cityAndCountry
stringByAddingPercentEncodingWithAllowedCharacters:allowed];
    encoded = [encoded stringByReplacingOccurrencesOfString:@" " withString:@"+"];

    if(self.useMock) {

        TestingHelper *helper = [TestingHelper shared];
        NSData *data = [helper forecastWithCityandstate:encoded];
        NSDictionary *forecast = nil;
        NSError *err = nil;
        if(data) {
            forecast = [NSJSONSerialization JSONObjectWithData:data options:0 error:nil];
        }
        else {
            forecast = @{@"StatusCode":@404, @"StatusMsg":@"No Such City/Country"};
        }

        if(completion) {
            completion(forecast, err);
        }

        return;
    }
```

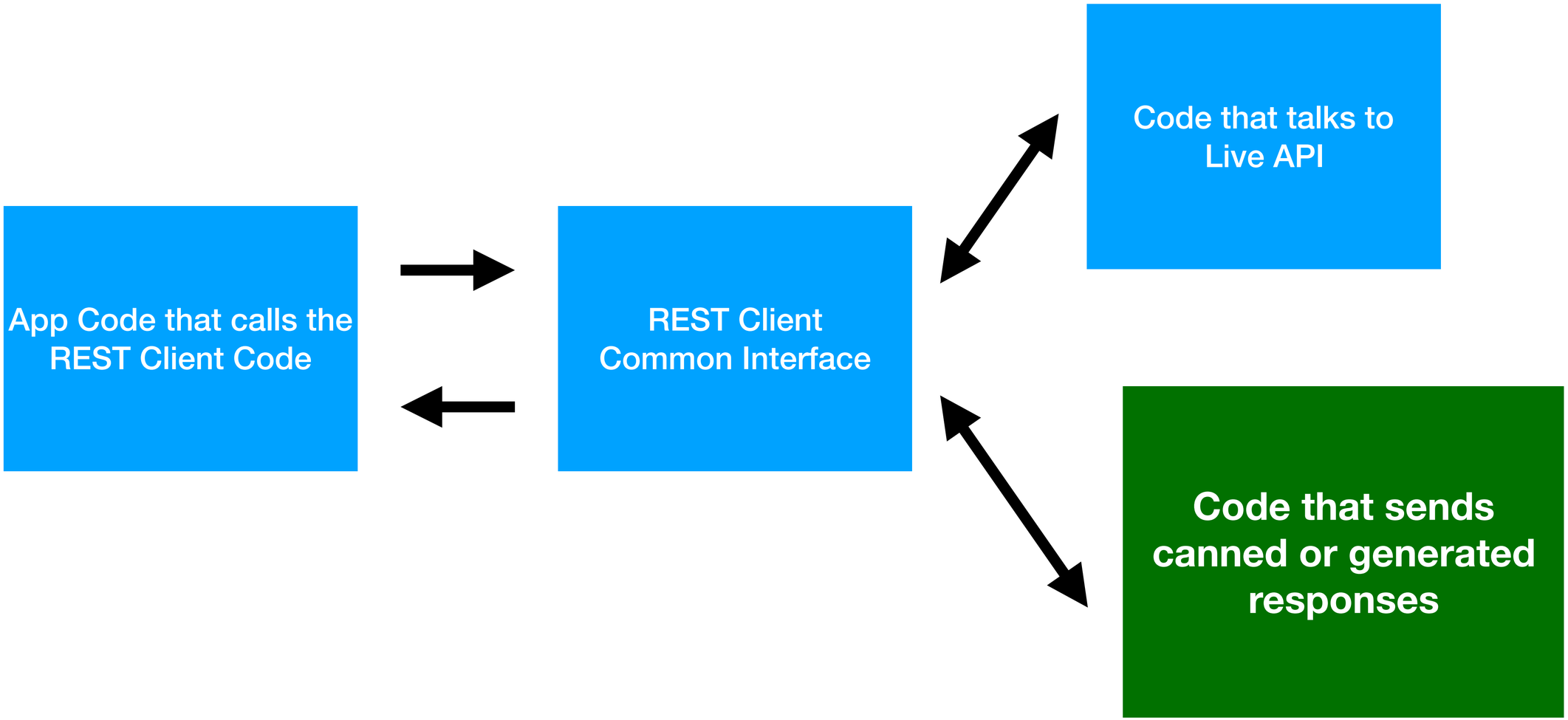
Method #2 - Embedded HTTP



Method #2 - Embedded HTTP

- Full control, but…
  - More to configure and manage
- Powerful, but…
  - You have to import someone’s framework
  - Presence of an open receiving socket trips security testers
  - You still have to maintain data & assets and you still have to script responses

Method #3 - Fake API



Method #3 - Fake API

```
@objc protocol WeatherApi {
    func forecast(for cityAndCountry:String, completion:@escaping ([String:Any]?, Error?) -> Void)
    func downloadIcon(for name:String, completion:@escaping ([String:Any], Error?) -> Void)
}


@objc class LiveRestAPI: NSObject, WeatherApi, URLSessionDelegate {


@objc class MockRestAPI: NSObject, WeatherApi {
```

Method #3 - Fake API

- Can call the *same methods*…
  - pass the *same types of parameters*…
  - get the *same type of result.*
- Can script the fake API to return specific errors
  - HTTP statuses and error messages
  - URL loading system NSError values (lost connection, server did not respond, etc)

Method #3 - Fake API

```
    func forecast(for cityAndCountry: String, completion: @escaping ([String:Any]?, Error?) -> Void) {
        var resultsDict:[String:Any] = [:]
        //construct path to JSON for cityAndCountry
        let bundle = Bundle.init(for: MockRestAPI.self)
        if let jsonPath = bundle.path(forResource: cityAndCountry, ofType: "json") {
            //look it up and return dictionary with response
            guard let data:NSData = try? NSData(contentsOfFile: jsonPath),
                let dict:[String:Any] = try! JSONSerialization.jsonObject(with: data as Data, options:
.mutableContainers) as? [String:Any]
            else {
                resultsDict["Status"] = 404
                completion(resultsDict, nil)
                return
            }
            resultsDict["Status"] = 200
            resultsDict["Data"] = dict
        }
        else {
            resultsDict["Status"] = 404
        }

        completion(resultsDict, nil)
    }
```

Method #3 - Fake API

```
func downloadIcon(for name: String, completion: @escaping ([String:Any], Error?) -> Void) {
    var resultsDict:[String:Any] = [:]
    //construct path to icon for name
    let bundle = Bundle.init(for: MockRestAPI.self)
    //look it up and return dictionary wtih response
    if let iconPath = bundle.path(forResource: name, ofType: "png") {
        //look it up and return dictionary wtih response
        guard let data:NSData = try? NSData(contentsOfFile: iconPath)
        else {
            resultsDict["Status"] = 404
            completion(resultsDict, nil)
            return
        }
        resultsDict["Status"] = 200
        resultsDict["Data"] = data
    }
    else {
        resultsDict["Status"] = 404
    }
    completion(resultsDict, nil)
}
```

Method #3 - Fake API

```
-(void)testGetMockForecastForNewYorkGet500Error {

    XCTestExpectation *expectation = [self expectationWithDescription:@"asynchronous
request"];
    __block NSDictionary *theForecast = nil;
    __block NSError *anythingHappened = nil;

    [self.restAPI loadProfileWithName:@"500_statuses"];
    [self.restAPI forecastFor:@"New York,USA" completion:^(NSDictionary *results, NSError
*err){
        theForecast = results;
        anythingHappened = err;
        [expectation fulfill];
    }];

    [self waitForExpectationsWithTimeout:10.0 handler:nil];

    XCTAssertNotNil(theForecast, @"Unable to get the forecast");
    XCTAssertNil(anythingHappened, @"Got error: %@", anythingHappened);
    XCTAssertTrue([@500 isEqualToNumber:theForecast[@"Status"]], @"Should be 200, not %@",
theForecast[@"Status"]);
}
```

Demo

Prototyping a REST API

- Work with server developers to define the API:
  - Endpoints (e.g. /forecast)
  - URI & POST body structure
  - Responses - status code, headers, response body
- Create JSON data with editor
- Add to testing bundles and code against them

API Mocking - Wrap-up

- Useful for testing, advance coding of UI, “playable demo”
- Not all that difficult
- Advantages over other methods
- Full control over data, including scriptable status & error codes
- By-product is testable design

Resources

- https://en.wikipedia.org/wiki/Representational_state_transfer
- https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol
- https://en.wikipedia.org/wiki/Dependency_injection
- https://stackoverflow.com/questions/2665812/what-is-mocking
- https://github.com/sciprojguy/4Cast/tree/swift_objc
- https://vimeo.com/265913162
- http://nshipster.com/nserror/