Building an Instagram Client in Swift

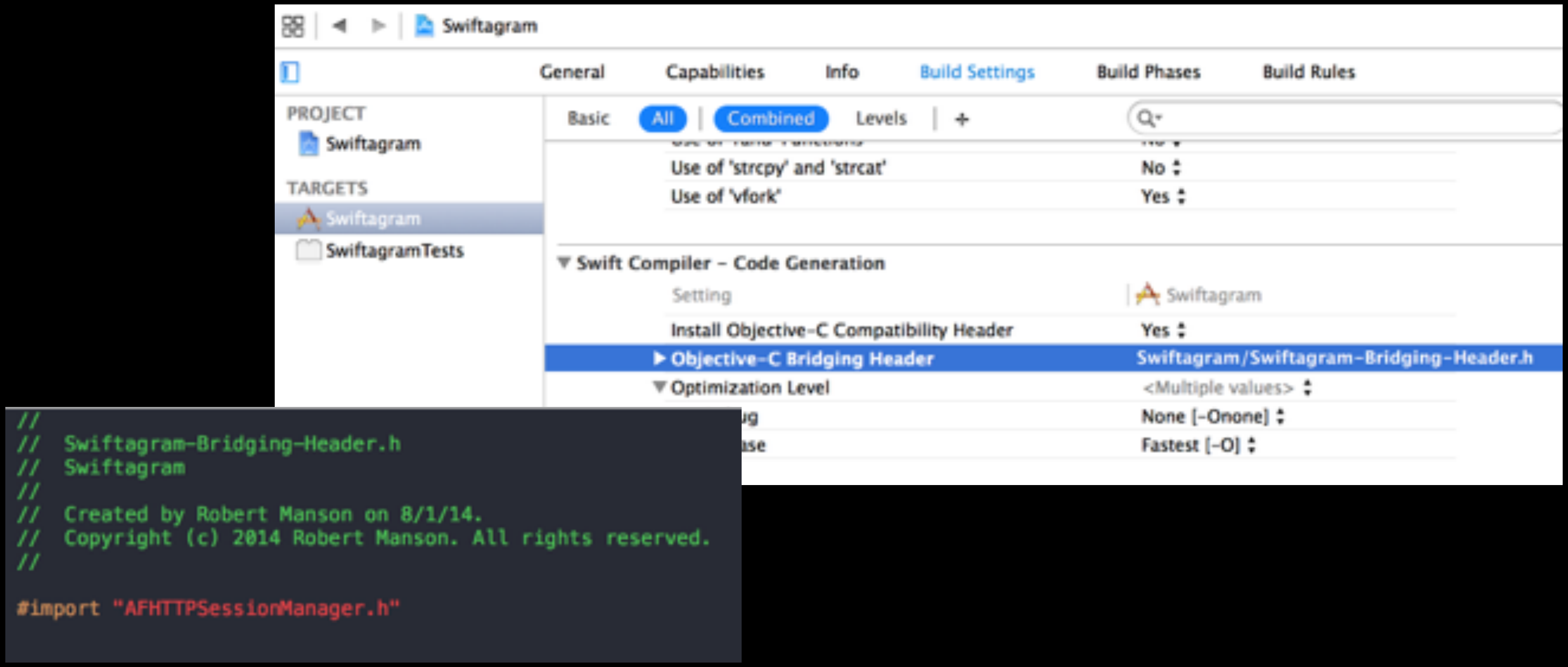
Robert Manson

AGOGO

- Bridging into Objective-C Code from your Swift app
- JSON Parsing
- Wrapping the API response in an enum

Bridging Objective-C Code

- Works kind of magically
- Don’t forget to set the bridging header for every target!



Use of 'strcpy' and 'strcat'	No
Use of 'vfork'	Yes

▼ Swift Compiler – Code Generation

Setting	Swiftagram
Install Objective-C Compatibility Header	Yes
▶ Objective-C Bridging Header	Swiftagram/Swiftagram-Bridging-Header.h
▼ Optimization Level	<Multiple values>
ug	None [-Onone]
ase	Fastest [-O]

```
//
//  Swiftagram-Bridging-Header.h
//  Swiftagram
//
//  Created by Robert Manson on 8/1/14.
//  Copyright (c) 2014 Robert Manson. All rights reserved.
//

#import "AFHTTPSessionManager.h"
```

Subclassing AFHTTPSessionManager

Turn this:

```
    - (id)init {
        self = [super initWithBaseURL:[NSURL URLWithString: @"https://
api.instagram.com/v1/"]];
    }
```

Into this:

```
    init() {
        let apiUrl = NSURL.URLWithString("https://api.instagram.com/v1/")
        super.init(baseURL: apiUrl)
    }
```

```
fatal error: use of unimplemented initializer
'init(baseURL:sessionConfiguration:)' for class
'Swiftagram.HttpClient'
```

- init(baseURL:sessionConfiguration:) is the designated initializer for AFHTTPSessionManager

- Designated initializer must call super’s designated initializer

- Swift apparently determines the which is the designated initializer in Objective-C code

Code

JSON Parsing

JSON Parsing

- Objective-C makes this fairly straightforward because of its message sending behavior and nil
- Strong types in Swift present new challenges since you are either dealing with [String: AnyObject] or [AnyObject]

How do I get the ID for the first blog?

```
let json: [String: AnyObject] = [
    "stat": "ok",
    "blogs": [
        "blog": [
            [
                "id" : 73,
                "name" : "Bloxus test",
                "needspassword" : true,
                "url" : "http://remote.bloxus.com/"
            ],
            [
                "id" : 74,
                "name" : "Manila Test",
                "needspassword" : false,
                "url" : "http://flickrtest1.userland.com/"
            ]
        ]
    ]
]
```

In Obj-C

```
dictionary[@"blogs"][@"blog"][0][@"id"]
```

* Don’t actually do this

In Swift?

```
if let blogsObj: AnyObject = json["blogs"] {
    if let blogs = blogsObj as? [String: AnyObject] {
        if let blogItems : AnyObject = blogs["blog"] {
            if let collection: [AnyObject] = blogItems as? [AnyObject] {
                let blogObj: AnyObject? = collection[0]
                if let blogInfo = blogObj as? [String: AnyObject] {
                    let idObj : AnyObject? = blogInfo["id"]
                    if let id = idObj as? Int {
                        println("Hating life right now: \(id)")
                    }
                }
            }
        }
    }
}
```

There must be a better way

We want something that

- Won’t throw a runtime error if what i’m looking for is not there
- Allows me to use option chaining
- Yields an optional
- Allow me to get at the original dictionary from anywhere in the JSON

Code

Other Approaches

- “Swift and JSON Reborn” by David Owens II
  - https://medium.com/swift-programming/b6f4f232e35e
  - Can’t get at the original dictionary

```
if let blogId = json[“blogs”]?[“blog”]?[0]?[“id”]?.number {
    println("blog ID: \(blogID)")
}
```

Other Approaches (2)

- “Parsing JSON in Swift” by Chris Eidhof
  - Co-Author of “Functional Programming in Swift”
  - Uses curried functions and operator overloading to parse into a struct
  - http://chris.eidhof.nl/posts/json-parsing-in-swift.html

```
mkBlog <*> int(dict,"id")
    <*> string(dict,"name")
    <*> bool(dict,"needspassword")
    <*> (string(dict, "url") >>= toURL)
```

Encapsulating the response

- Endpoint agnostic wrapper that tells me if the call was successful
- If I succeed, give me a dictionary of the JSON
- If I fail, give me an NSError
- Perfect use case for an Enum since they can hold associated values

Code

Thanks!

Robert Manson
@litso
github.com/litso