SwiftCon China 2016

www.swiftconchina.com

一个 Swift 项目
网络层的变迁

陈乘方

Swift 项目介绍

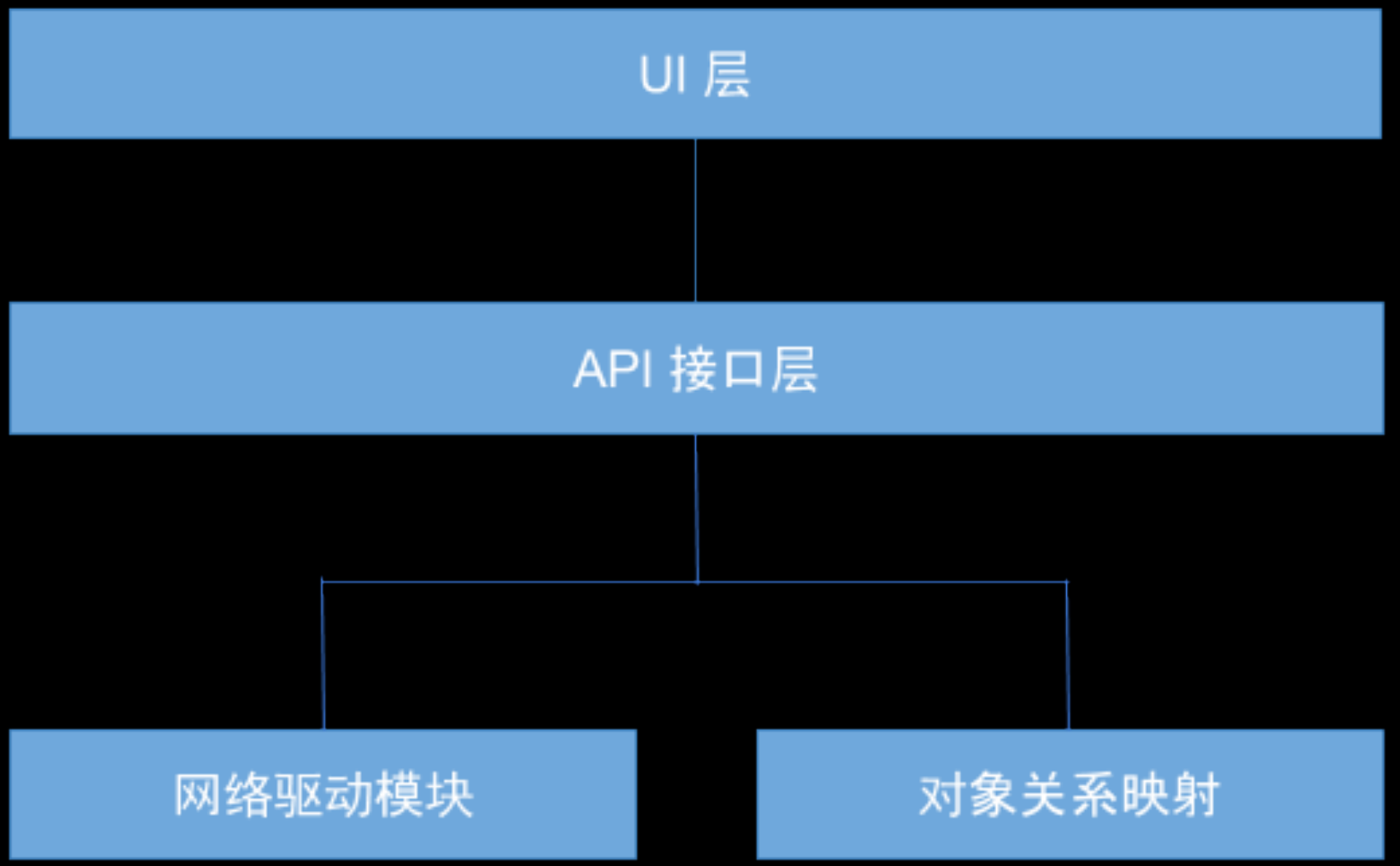
- 项目内部代号 Daenerys，App 正式的名字「ENJOY-精选限量美食」

- 始于 2014 年 10 月份，纯 Swift 项目。历经了 Swift 1.1 至今的语言版本变化

- 2014-11-26 提交 1.0 版本审核， 2015-01-15 App Store 正式上架

什么变化?

Swift 化！



ENJOY 网络层结构

主要变化

- 第三方框架的替换
- 网络相关 Model 的变化
- 回调方式改变
- 支持链式调用形式处理网络返回结果
- API 接口层独立为一个 framework

```swift
    func fetchChannelList(success: ((channelList: [RBDChannel]) -> Void), failure: ((error: NSError) -> Void)) {
        sendGetWithRelativeUrl("3/deal/list_channel.json", params: [:], success: { (responseObject) -> Void in

            if let responseArray = responseObject as? [[String: AnyObject]] {
                var error: NSError?
                var resultArray: [RBDChannel] = []

                for channelDic in responseArray {
                    let channel = MTLJSONAdapter.modelOfClass(RBDChannel.self, fromJSONDictionary: channelDic, error: &error)
as! RBDChannel
                    if error != nil {
                        failure(error: self.p_createServerDataFormatError())
                        return
                    }

                    resultArray.append(channel)
                }

                success(channelList: resultArray)

            } else {
                failure(error: self.p_createServerDataFormatError())
            }
        }) { (error) -> Void in
            failure(error: error)
        }
    }
```

```
func fetchChannelList(completionHandler: (result: APIResult<[ChannelModel]>) -> Void ) {
    SCFNetworkKit.requestAPI(.Get, APIName: "3/deal/list_channel.json").response { (result) -> Void in
        completionHandler(result: result
            .flatMap(jsonObjectToResponseDicArray)
            .flatMap(responseDicArrayToAPIModelArray)
        )
    }
}
```

所用 Swift 特性

- Enum Associated Values
- Generic
- Protocol Extension

变迁过程

围绕着枚举进行的网络层渐进改造

最初的 API 接口层特点

- 根据业务需求对网络驱动框架做了简单封装
- 使用 extension 对一个单例的 API 接口管理类扩展方法，来封装每一个 API 接口
- 使用 success 和 failure 的双 closure 回调形式
- 每一个 API 接口对应的封装方法内都有各自对应的数据处理过程

```
extension RBDNetworkKit {
  func functionName(
    success: ((result: Model)-> Void),
    failure: ((error: NSError) -> Void)) {

      sendGetWithRelativeUrl(“api.json",
        params: [:],
        success: {
          (responseObject) -> Void in
            // Model 处理
          success(result: model)
        }) { (error) -> Void in
          failure(error: error)
        }
    }
}
```

网络驱动框架替换

- AFNetworking 变为 Alamofire
- 基本无痛的切换

回调方式改变

- success + failure 的双回调形式变为 completion 单回调
- 核心是定义了一个 Swift 枚举：APIResult
- 参照 Alamofire 修改了 API 接口层的网络调用方式

```
enum APIResult<T> {
    case Success(T)
    case Failure(NSError, AnyObject?)
}
```

```
func functionName(
        success: (model: Model) -> Void,
        failure: (error: NSError) -> Void
)
```

```
enum APIResult<T> {
    case Success(T)
    case Failure(NSError, AnyObject?)
}
```

```
func functionName(completionHandler:
        (result: APIResult<Model>) -> Void
)
```

公共 framework

- 随业务发展及新项目的启动，一些公共逻辑拆分出来形成了一个公共的 framework
- 网络请求相关的逻辑也放到了这个 framework

```
enum APIResult<T> {
    case Success(T)
    case Failure(NSError, AnyObject?)
}
```

```
public enum APIResult<T> {
    case Success(T)
    case Failure(NSError, AnyObject?)
}
```

```
extension RBDNetworkKit {
  func functionName(
    success: ((result: Model)-> Void),
    failure: ((error: NSError) -> Void)) {

      sendGetWithRelativeUrl(“api.json",
        params: [:],
        success: {
          (responseObject) -> Void in
            // Model 处理
          success(result: model)
        }) { (error) -> Void in
          failure(error: error)
        }
    }
}
```

```
extension SCFNetworkKit {
  func functionName(completionHandler:
    (result: APIResult<Model>) -> Void) {

      SCFNetworkKit
        .requestAPI(.Get,
                APIName: "api.json",
              paramsDic: [:])
          .response { (result) -> Void in
              // 处理结果
          }
  }
}
```

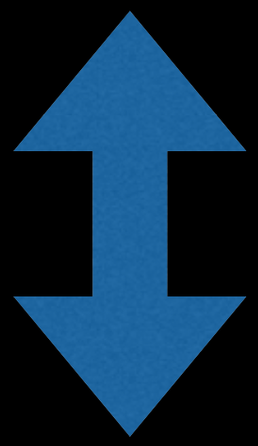
网络 Model 变化

- ObjectMapper 代替了 Mantle
- 去掉命名前缀
- Struct 类型替换了 Class 类型
- 尽可能减少 optional 类型的属性

ObjectMapper 与 Mantle 对比

- ObjectMapper 使用 Swift 编写，Mantle 使用 Objective-C 编写
- ObjectMapper 支持转换为原生的 Swift 类型
- ObjectMapper 支持 Struct
- ObjectMapper 的映射关系写法更安全

```
class func JSONKeyPathsByPropertyKey()
                -> [NSObject : AnyObject]! {
    return [
      “channelID": "channel_id",
      "channelName": "channel_name",
    ]
}
```



```
mutating func mapping(map: Map) {
    channelID <- (map["channel_id"], Int64Transform())
    channelName <- map["channel_name"]
}
```

ObjectMapper 扩展

- 有些时候 Model 里的一些属性是不能为空的，最常见的就是各种 ID
- 通过对 ObjectMapper 扩展来支持这一特性

```
protocol NonnilMappable: Mappable {

    var nonnilMapProperties: [String] { get }

    func shouldReturnNil(map: Map) -> Bool

}

extension NonnilMappable {

    func shouldReturnNil(map: Map) -> Bool {
        for property in nonnilMapProperties {
            if map[property].currentValue == nil {
                return true
            }
        }

        return false
    }

}
```

网络结果的链式调用

- Swift 中最典型的链式调用就是 optional chain
- optional 类型也是通过枚举实现
- 链式调用的关键在于调用方法返回与自身相同的类型

```
public enum APIResult<T> {
    case Success(T)
    case Failure(NSError, AnyObject?)

}
```

```
public enum APIResult<T> {
    case Success(T)
    case Failure(NSError, AnyObject?)

    public func flatMap<U>(@noescape transform: T throws -> U? ) rethrows
          -> APIResult<U> {
        switch self {
        case let .Failure(error, obj):
            return .Failure(error, obj)
        case let .Success(value):
            guard let newValue = try transform(value) else {
                return .Failure(
                    SCFNetworkKit.createAPIServerDataFormatError(),
                    nil)
            }
            return .Success(newValue)
        }
    }

    public func map<U>(@noescape transform: T throws -> U) rethrows
          -> APIResult<U> {
        return try flatMap { try transform($0) }
    }
}
```

数据处理通用组件

- 对数据处理过程的抽象
- 提供通用函数作为组件
- 组件函数的参数和返回值符合 flatMap 参数定义

```
func jsonObjectToResponseDic(obj: AnyObject)
    -> [String: AnyObject]? {
    return obj as? [String: AnyObject]
}
```

```swift
public protocol APIModelConvertible {
    static func toModel(dic: [String: AnyObject]) -> Self?
}

extension APIModelConvertible where Self: Mappable {
    static func toModel(dic: [String : AnyObject]) -> Self? {
        return Mapper().map(dic)
    }
}


func responseDicToAPIModel<T: APIModelConvertible>
    (dic: [String: AnyObject]) -> T? {
    return T.toModel(dic)
}
```

APIResult 实用扩展

- APIResult 是一个封装
- UI 层每次拿到 APIResult 后都需要解包处理

```
SCFNetworkKit.sharedInstance.functionNameOfAPI {
    (result) -> Void in
        switch result {
            case let .Success(model):
                // 成功处理
                break
            case let .Failure(error, _):
                // 失败处理
                break
        }
}
```

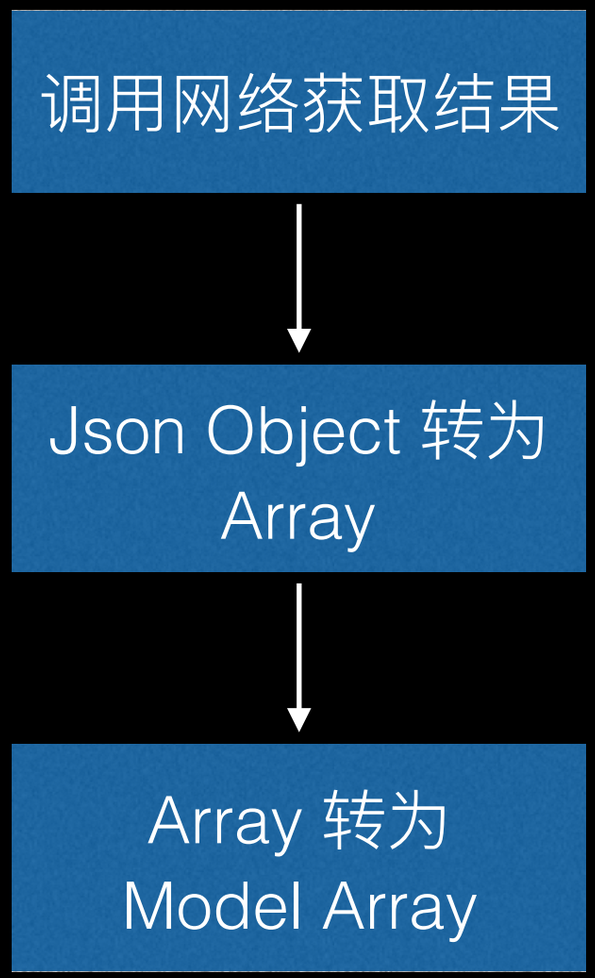
```
public enum APIResult<T> {

    .. ..

     public func success(@noescape success: (value: T) -> Void)
         -> APIResult<T> {
         switch self {
         case let .Success(value):
             success(value: value)
         default:
             break
         }
         return self
     }
     public func failure(@noescape failure:
         (error: NSError, obj: AnyObject?) -> Void) -> APIResult<T> {
         switch self {
         case let .Failure(error, obj):
             failure(error: error, obj: obj)
         default:
             break
         }
         return self
     }
}
```

```
SCFNetworkKit.sharedInstance.functionNameOfAPI {
    (result) -> Void in
        result
            .success({ (model) -> Void in
                // 成功处理
            })
            .failure({ (error, obj) -> Void in
                // 失败处理
            })
}
```

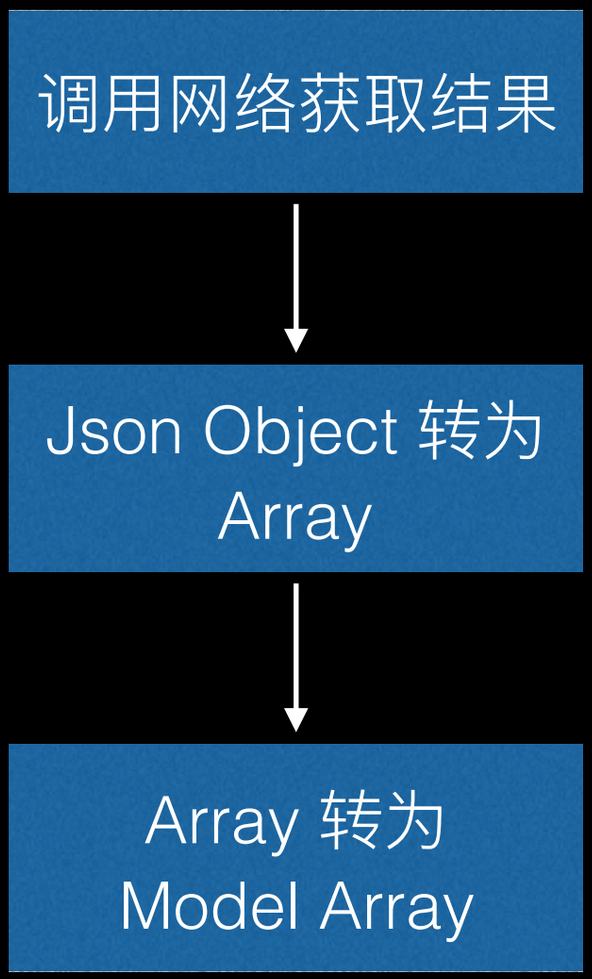
举例

ChannelList 接口

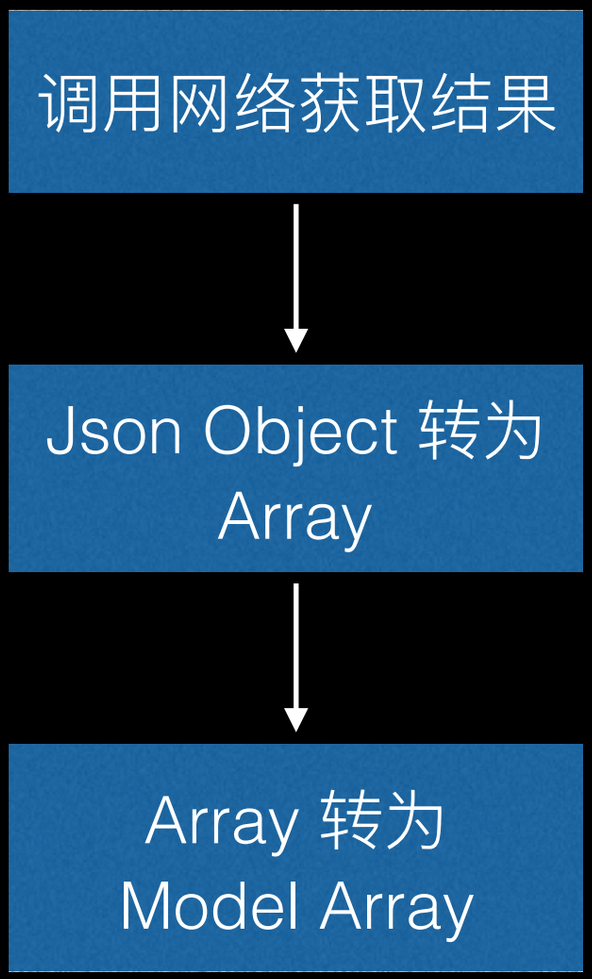
```
struct ChannelModel: APIModelConvertible, NonnilMappable {
    var channelID: Int64 = 0
    var channelName: String = ""

    init?(_ map: Map) {

        if shouldReturnNil(map) {
            return nil
        }
    }

    mutating func mapping(map: Map) {
        channelID <- (map["channel_id"], Int64Transform())
        channelName <- map["channel_name"]
    }

    var nonnilMapProperties: [String] {
        return [
            “channel_id", "channel_name",
        ]
    }
}
```




```
func fetchChannelList(completionHandler:
  (result: APIResult<[ChannelModel]>) -> Void ) {

}
```




```
func fetchChannelList(completionHandler:
   (result: APIResult<[ChannelModel]>) -> Void ) {
    SCFNetworkKit
          .requestAPI(.Get,
                APIName: “3/deal/list_channel.json”)
          .response { (result) -> Void in
              // 网络请求结果处理
          }
    }
```




```
func fetchChannelList(completionHandler:
  (result: APIResult<[ChannelModel]>) -> Void ) {
   SCFNetworkKit
         .requestAPI(.Get,
               APIName: “3/deal/list_channel.json”)
         .response { (result) -> Void in
            // 网络请求结果处理
            completionHandler(result: result
              .flatMap(jsonObjectToResponseDicArray)
         )
      }
}
```

调用网络获取结果

↓

Json Object 转为 Array

↓

Array 转为 Model Array

```
func fetchChannelList(completionHandler:
  (result: APIResult<[ChannelModel]>) -> Void ) {
   SCFNetworkKit
         .requestAPI(.Get,
               APIName: “3/deal/list_channel.json”)
         .response { (result) -> Void in
            // 网络请求结果处理
            completionHandler(result: result
              .flatMap(jsonObjectToResponseDicArray)
              .flatMap(responseDicArrayToAPIModelArray)
         )
      }
  }
```

最终结果

- 提高了网络模块相关代码的复用率
- 更直观方便的 API 接口封装方式
- 减少了封装 API 接口的代码量
- 减少 Bug 的产生

2015/09/01

```
15:37:21 ...develop/iOS/iOS-Daenerys 7c225cb ✘ ★
$ cloc Utils/Network
      15 text files.
      15 unique files.
       0 files ignored.

http://cloc.sourceforge.net v 1.64  T=0.09 s (159.3 files/s, 45373.5 lines/s)
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
Swift                           15            942            739           2591
-------------------------------------------------------------------------------
SUM:                            15            942            739           2591
-------------------------------------------------------------------------------
```

2016/03/13

```
15:19:25 ...iOS-Daenerys/Utils/NetworkKit develop ✔
$ cloc .
      19 text files.
      19 unique files.
       0 files ignored.

http://cloc.sourceforge.net v 1.64  T=0.09 s (202.8 files/s, 38043.6 lines/s)
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
Swift                           19            900            646           2018
-------------------------------------------------------------------------------
SUM:                            19            900            646           2018
-------------------------------------------------------------------------------
```

最后

- 如果你喜欢 Swift
- 如果你是冰与火之歌的脑残粉
- 如果你乐于探索新技术并且有代码洁癖
- 欢迎投简历到 chenchengfang@ricebook.com

Thank You