从零开始打造一个 Swift 网络框架

2016-05-19 08:27 编辑：cocopeng 分类：Swift 来源：伯乐在线 0 576

Swift

招聘信息：Web后端高级开发工程师

说起网络框架，大家第一时间就会想到 AFNetworking、Alamofire 这些业内响当当的作品，有的老鸟也会适当伤感一下曾经用的 ASI 。这些框架都有一个共同点——功能都很复杂，很齐全，而我们往往只能用到很小很小的一个部分。

事实上，咱们做 App 的时候，绝大多数时候对网络的需求都是收发 GET/POST 请求。就这样来看，根据需求来造个属于自己的轮子，似乎也是个不错的选择。尤其是现在苹果提供的 NSURLSession 已经非常强大，基于原生的 SDK 来做一个自己的框架，其实是很容易的。

根据这个思想，我之前撸了一个简单的网络库 AaHTTP，在工作的项目里重度用了一段时间也没有遇到什么特别的问题。

现在我们就来一步步看看如何做一个属于自己的简单的网络框架。

发送请求的步骤分析

要发送一个请求，分为如下步骤：

1. 如果携带的参数是 GET 类型，则将参数进行 URL encode（转化为 y1=x1&y2=x2的形式），追加到原始 url 的后面。如果参数是 POST 类型，则 URL 不变。
2. 用最新的 URL 生成一个 NSMutableURLRequest 的对象
3. 如果参数是 POST 的情况，设置 Content-Type 为 application/x-www-form-urlencoded, 并将参数进行 URL encode，并添加到 body 中。
4. 使用 NSURLSession 发送该请求

URL encode时，需要对特殊字符进行转义。

定义发送请求的接口

根据上面的步骤，我们不难一步到位的实现发送请求，新建一个 AaNet.swift （名字您随意），并声明我们的类方法：

```
class AaNet: NSObject {
class func request( method : String = "GET",url : String ,form : Dictionary = [:],succes
}
func buildParams(parameters: [String: AnyObject]) -> String {
return ""
}
}
```

我们首先声明了两个函数，request 函数接受的参数依次是：

```
method: 请求类别
url: 目标地址
form: 参数表
success: 成功的回调，类型为(data:NSData?) -> Void
fail: 失败的回调,类型为(error : NSError?) -> Void
```

热门资讯


App Store应用审查缩至24小时
点击量 9798


知道这20个正则表达式，能让你少写1,000
点击量 9019


Swift 3.0 预告：将 Objc 库转换成更符合
点击量 7475


苹果最刁钻的33个面试题，有种就来挑战
点击量 7162


iOS网络层架构设计分享
点击量 6995


移动端数据库新王者：realm
点击量 5813


苹果：6月1日后所有应用必须支持IPv6-only
点击量 4823


为设计师精心打造的14款进阶移动端应用
点击量 4810


一个计算机专业毕业生工作5年后的困惑
点击量 3867


2016年5月TIOBE编程语言排行榜：Ruby创
点击量 3806

综合评论

这个666666，这个是不是说以后直接在电脑上玩手机游戏了？不用怕手机发 lirch 评论了 谷歌送惊喜：Chrome浏览器可直接运行安卓应用...

swift式的写法 就是把model写成Struct jjhouse 评论了 用RxSwift实现一个UITableView...

我们的工资都是被这些砖家“增长的”。 冰宫无凉 评论了 国家统计局说IT行业人均年薪112042元，你拖后...

哦哦 iosjob2015 评论了 iOS 三种录制视频方式...

第二个函数 buildParams, 输入一个字典，返回一个字符串。很容易想到就是我们用来做 url encode 的函数。

建议大家写代码前，都先写出主要函数的声明和对应的参数、返回值的类型。这其实就是一种最基本的架构工作

实现发送请求

现在按照之前的分析，我们来实现请求发送的逻辑：

```
class func request( method : String = "GET",url : String ,form : Dictionary = [:],succe
var innerUrl = url
if method == "GET"{
innerUrl += "?" + AaNet().buildParams(form)
}
let req = NSMutableURLRequest(URL: NSURL(string: innerUrl)!)
req.HTTPMethod = method
if method == "POST" {
req.addValue("application/x-www-form-urlencoded", forHTTPHeaderField: "Content-Type")
print("POST PARAMS (form)")
req.HTTPBody = AaNet().buildParams(form).dataUsingEncoding(NSUTF8StringEncoding)
}
let session = NSURLSession.sharedSession()
print(req.description)
let task = session.dataTaskWithRequest(req) { (data, response, error) -> Void in
if error != nil{
fail(error: error)
print(response)
}else{
if (response as! NSHTTPURLResponse).statusCode  == 200{
success(data : data)
}else{
fail(error: error)
print(response)
}
}
}
task.resume()
}
```

整个流程很直观，虽然 GET 参数和 POST 参数处理的位置不同，但都是用我们的 url encode 函数 buildParams 来操作的。区别是 GET 请求的话，处理完后直接 append 到 url 后面，而 POST 需要用 UTF8 encode 一下，放在 request 的 body 里。

然后用 NSURLSession 的默认 session： sharedSession() 来发送请求，并在回调里判断 statusCode 以及 error 对象是否为 nil 来判断请求是否为空，来分别调用我们的 success 回调或 fail 回调。

实现 URL encode

现在我们来实现 buildParams，大体的步骤为：

encode：

1. 把输入字典转换为键值对的数组。[(Key,Value)]

2. 对于每一个 (key,value)，执行：

2.1 对 key 进行转义，得到 key'

2.2 检查 value 的类型，如果是简单的值，则对其进行转义，得到 value'。并将 (key' , value') 输出到结果数组中。

2.3 如果 value 是数组，则用当前的 key 和 value 中的每一个元素组成 tuple： [(key,subValue)], 递归执行步骤2。

2.4 如果 value 是字典，也先把 value 对应的字段转化为键值对数组，但是 key 的形式为 key[subKey], 前面是 key 是当前的 key，subKey 代表 value 对应的字典中的 key。得到键值对数组后，递归执行步骤2。

3. 步骤2执行完毕后，我们会得到一个一维的、并且 key 和 value 都被转义过的键值对数组 [(key,value)]，然后我们将其转换为 key1=value1&key2=value2&...keyN=valueN 的形式返回。

仔细感受一下，步骤2是不是有一个 flat 的过程。

我们先实现转义：

```
func escape(string: String) -> String {
let legalURLCharactersToBeEscaped: CFStringRef = ":&=;+!@#$()',*"
return CFURLCreateStringByAddingPercentEscapes(nil, string, nil, legalURLCharactersToBe
}
```

没啥技术含量，可直接抄去用。然后根据我们上面的分析，实现 URL encode：

```
func buildParams(parameters: [String: AnyObject]) -> String {
var components: [(String, String)] = []
for key in Array(parameters.keys).sort() {
let value: AnyObject! = parameters[key]
components += self.queryComponents(key, value)
}
return (components.map{"($0)=($1)"} as [String]).joinWithSeparator("&")
}
func queryComponents(key: String, _ value: AnyObject) -> [(String, String)] {
var components: [(String, String)] = []
if let dictionary = value as? [String: AnyObject] {
for (nestedKey, value) in dictionary {
components += queryComponents("(key)[(nestedKey)]", value)
}
} else if let array = value as? [AnyObject] {
for value in array {
components += queryComponents("(key)", value)
}
} else {
components.appendContentsOf([(escape(key), escape("(value)"))])
}
return components
}
```

我们用了一个辅助函数 queryComponent 来表达步骤2这个递归过程。

至此，我们就完成了请求的封装，这个部分完整的代码在这里

现在我们就可以用它来发送请求了，比如我们想通过 bing 网页词典来查询 joepardize 这个单词的意思：

```
AaNet.request("GET", url: "http://cn.bing.com/dict/", form: ["q":"jeopardize"], success:
print(String(data: data!, encoding: NSUTF8StringEncoding))
}) { (error) in
}
```

返回：（这里没有对结果进行 parse, 这个不属于本文的内容

```
**Optional("//jeopardize - ****必应**** Dictionary//
```

更优雅的接口和适配器模式

显然，目前的接口并不友好，封装也很低级。对于移动应用的网络开发而言，还有几个基本的需求没有被覆盖：

- 默认的主机名： 我们的 app 一般的后台就一个域名，如果我们每次发一个请求都要敲一遍域名那真的太蛋疼了。
- 默认的参数列表： 很多参数是基本每个请求都要带的，比如 app 的版本，用户设备的语言等等。
- 更加简短并让人一看就懂得函数调用。
- 参数可缺省
- 错误处理可缺省

要实现上述的需求，我们有两条路可以走：

- 在 AaNet 内部加上对应的逻辑，然后对之前的 request 做各种函数重载来实现。
- 做一个新的模块，实现上述功能，但底层的数据发送调用 AaNet, AaNet 代码不变。

凭直觉来看，似乎应该选择第二个方案，首先上面的需求可能是多变的，但 AaNet 目前完成的功能是基本不会变的（除非 HTTP 协议的标准改变），变化的和不变的应该分开。其次是我们在将来有可能遇到 AaNet 不能满足我们的需求，需要采用一些更加成熟的框架（e.g. AFNetworking 等）的时候，迁移的成本要最低的话，用一个中间层把我们的代码和 AaNet 隔开是个很不错的选择。

这个思想在设计模式中叫做**适配器模式**， 我们新开一个 AaHTTP （名字任意）类来处理上述的需求，在底层调用 AaNet 来实现请求的发送。 然后在代码里调用 AaHTTP 的方法来完成业务逻辑，这样，即便某一天我们要需要替换网络通信的框架，也只是需要在 AaHTTP 内部的实现上修改 AaNet 为其他实现即可，不需要修改其他代码。 这里的 AaHTTP 就是一种典型的适配器。

实现 AaHTTP

比起 AaNet， AaHTTP 的实现是很简单的，主要都是一些设计层面的东西。

方便区别 GET 和 POST， 用字符串肯定是不明智的，我们增加一个 enum:

```
enum RequestMethod{
case Post
case Get
}
```

成员变量什么的就不用一一列举了，大家可以直接查看该文件完整的源代码。 这里看一下对外暴露的4个方法

为了实现链式调用，每个方法返回的都是自身

```
func fetch(url : String) -> AaHTTP{
setDefaultParas()
curUrl = "(hostName)(url)"
self.method = .Get
return self
}
func post(url : String) -> AaHTTP{
setDefaultParas()
curUrl = "(hostName)(url)"
self.method = .Post
return self
}
func paras(p : [String:AnyObject]) -> AaHTTP{
_ = p.reduce("") { (str, p) -> String in
parameters[p.0] = p.1
return ""
}
return self
}
func go(success : String -> Void, failure : NSError?->Void){
var smethod = ""
if method == .Get{
smethod = "GET"
}else{
smethod = "POST"
}
AaNet.request(smethod, url: curUrl, form: parameters, success: { (data) -> Void in
print("request successed in (self.curUrl)")
let result = String(data: data!, encoding: NSUTF8StringEncoding)
success(result!)
}) { (error) -> Void in
print("request failed in (self.curUrl)")
failure(error)
}
}
```

fetch 和 post 分别生成 GET 和 POST 请求，paras 方法设置参数，go 方法进行实际请求操作。

现在，我们可以这样来发送网络请求：

```
aht.shareInstance.fetch("http://yahoo.com").go({ (result) in print(result) }) { (error)
```

如果有参数的话：

```
aht.shareInstance.fetch("http://cn.bing.com/dict/").paras(["q":"jeopardize"]).go({ (resu
```

通过该类内部的 hostname 属性，即可实现缺省的主机名。

结语

至此，我们就完成了一个最简单、但足以应付绝大多数网络请求的框架，或者也可以基于此走得更远，比如：

- 尝试管理多个 NSURLSession
- 尝试实现文件的下载与上传
- 尝试集成常见的 restful api authentication 的功能，比如 BCE的鉴权机制

微信扫一扫

订阅每日移动开发及APP推广热点资讯

公众号：CocoaChina

我要投稿 收藏文章 分享到：

1

上一篇：静态类型的 NSUserDefaults

相关资讯

- 【译】“错误”的使用 Swift 中的 Extension
- 静态类型的 NSUserDefaults
- Swift 3.0 预告：更符合 Swift 风格的调试标识
- Swift 3.0 预告：将 Objc 库转换成更符合 Swift 语法风格
- 探索 Swift 中的 MVC-N 模式
- Swift 实现俄罗斯方块详细思路解析（附完整项目）
- Swift 3.0首个开发者预览版将在5月12日发布
- Swift 中关于”??”操作符一些有意思的事情
- 10个惊艳的Swift单行代码
- 为什么Swift中应该避免使用guard语句

我来说两句

您还没有登录！请 登录 或 注册

所有评论（0）