Alamofire 入门

一，Alamofire的说明与配置

1，什么是Alamofire

（1）Alamofire的前身是AFNetworking。AFNetworking是iOS和OS X上很受欢迎的第三方HTTP网络基础库。

（2）其实AFNetwork的前缀AF便是Alamofire的缩写。

（3）Swift发布后，AFNetworking的作者又用Swift语言写了个相同功能的库，这便是Alamofire。

（4）Alamofire本质是基于`URLSession`，并做了封装。使用Alamofire可以让我们网络请求相关代码（如获取数据，提交数据，上传文件，下载文件等）更加简洁易用。

关于*Cookie*：

Alamofire是基于URLRequest封装的，所以Cookie会自动保存，就和浏览器请求是一个效果。而且网站Set_cookie多久，本地的Cookie就多久，每次请求的时候都会自动带上cookie，直到过期。（所以像登陆session这些的都不用我们手动去处理）

2，Alamofire的功能特性：

（1）链式的请求/响应方法

（2）URL / JSON / plist参数编码

（3）上传类型支持：文件（File）、数据（Data）、流（Stream）以及MultipartFormData

（4）支持文件下载，下载支持断点续传

（5）支持使用NSURLCredential进行身份验证

（6）HTTP响应验证

（7）TLS Certificate and Public Key Pinning

（8）Progress Closure & NSProgress

3，Alamofire的安装与配置

（1）从GitHub上下载最新的代码

（2）最后，在需要使用Alamofire的地方import进来就可以了： import Alamofire

二，使用Alamofire进行数据请求

1，以GET请求为例

（1）不带参数，不带结果处理

```
Alamofire.request("https://httpbin.org/get")
```

（2）带参数，不带结果处理

```
Alamofire.request("https://httpbin.org/get", parameters: ["foo":"bar"])
```

（3）带参数，也带结果处理（这里以返回结果为json格式的为例）

```
Alamofire.request("https://httpbin.org/get", parameters: ["foo":"bar"])
.responseJSON { responsein
print(response.request)// original URL request
print(response.response)// URL response
```

```
print(response.data)// server data
print(response.result)// result of response serialization
ifletJSON= response.result.value {
print("JSON: \(JSON)")//具体如何解析json内容可看下方“响应处理”部分
}
}
```

，响应处理（Response Handling）

1）除了上面样例使用的responseJSON（处理json类型的返回结果）外，Alamofire还提供了许多其他类型的响应处理方法：

```
response()
responseData()
responseString(encoding: NSStringEncoding)
responseJSON(options: NSJSONReadingOptions)
responsePropertyList(options: NSPropertyListReadOptions)
```

2）Response Handler

```
Alamofire.request("https://httpbin.org/get", parameters: ["foo":"bar"])
.response { responsein
print("Request: \(response.request)")
print("Response: \(response.response)")
print("Error: \(response.error)")
if  let   data = response.data, letutf8Text =String(data: data, encoding: .utf8) {
print("Data: \(utf8Text)")
}
}
```

（3）Response Data Handler

```
Alamofire.request("https://httpbin.org/get", parameters: ["foo":"bar"])
.responseData { responsein
debugPrint("All Response Info: \(response)")
if let data = response.result.value,letutf8Text =String(data: data,
encoding: .utf8) {
print("Data: \(utf8Text)")
}
}
```

（4）Response String Handler

```
Alamofire.request("https://httpbin.org/get", parameters: ["foo":"bar"])
.responseString { responsein
print("Success: \(response.result.isSuccess)")
print("Response String: \(response.result.value)")
}
```

（5）Response JSON Handler

使用responseJSON 方法的话，JSON数据会被自动转化为 Dictionary或Array。假设我们返回的json数据格式如下：

```
[
{
"name": "hangge",
```

```
"phones": [
{
"name": "公司",
"number": "123456"
},
{
"name": "家庭",
"number": "001"
}
]
},
{
"name": "big boss",
"phones": [
{
"name": "公司",
"number": "111111"
}
]
}
]
```

使用 responseJSON 自动解析 json 数据：

```
Alamofire.request("http://www.hangge.com/jsonData.php")
.responseJSON { responsein
switch response.result.isSuccess {
case  true:
//把得到的 JSON 数据转为数组
if let items = response.result.valueas?NSArray{
//遍历数组得到每一个字典模型
for dict in  items{
print(dict)
}
}
casefalse:
print(response.result.error)
}
}
```

（6）同样也支持链式的返回结果处理

```
Alamofire.request("https://httpbin.org/get")
.responseString { responsein
print("Response String: \(response.result.value)")
}
.responseJSON { responsein
print("Response JSON: \(response.result.value)")
}
```

3，请求类型（HTTP Methods）

除了上面使用的.Get类型（不指定的话，默认都是使用Get请求）。Alamofire还定义了许多其他的HTTP 方法（HTTP Medthods）可以使用。

```
public enum  HTTPMethod:String{
caseoptions ="OPTIONS"
caseget="GET"
casehead    ="HEAD"
casepost    ="POST"
caseput     ="PUT"
casepatch   ="PATCH"
casedelete  ="DELETE"
casetrace   ="TRACE"
caseconnect ="CONNECT"
}
```

比如要使用POST请求，把Alamofire.request第二个参数做修改即可：

```
 Alamofire.request("http://httpbin.org/post", method: .post)
```

4，请求参数（Parameters）

（1）使用GET类型请求的时候，参数会自动拼接在url后面

```
Alamofire.request("https://httpbin.org/get", parameters: ["foo":"bar"])
//https://httpbin.org/get?foo=bar
```

（2）使用POST类型请求的时候，参数是放在在HTTP body里传递，url上看不到

```
letparameters:[String:Any] = [
"foo":"bar",
"baz": ["a", 1],
"qux": [
"x": 1,
"y": 2,
"z": 3
]
]
Alamofire.request("https://httpbin.org/post", method: .post, parameters: parameters)
// HTTP body: foo=bar&baz[]=a&baz[]=1&qux[x]=1&qux[y]=2&qux[z]=3
```

5，参数编码方式（Parameter Encoding）

除了默认的方式外，Alamofire还支持URL、JSON、PropertyList以及自定义格式方式编码参数。

比如我们想要把一个字典类型的数据，使用json格式发起POST请求：

```
let parameters:[String:Any] = [
"foo": [1,2,3],
"bar": [
"baz":"qux"
]
]
Alamofire.request("https://httpbin.org/post", method: .post, parameters: parameters,
encoding:JSONEncoding.default)
// HTTP body: {"foo": [1, 2, 3], "bar": {"baz": "qux"}}
```

服务端php页面可以这么取得发送过来的JSON数据：
$postdata= json_decode(file_get_contents("php://input"),TRUE);
$foo=$postdata["foo"];
foreach($fooas$item){
echo$item."|";
}
//输出：1|2|3|
6，支持自定义Http头信息（HTTP Headers）
let headers:HTTPHeaders= [
"Authorization":"Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==",
"Accept":"application/json"
]
Alamofire.request("https://httpbin.org/headers", headers: headers)
.responseJSON { responsein
debugPrint(response)
}

三，判断数据请求是否成功，并做相应的处理
在请求响应对象之前调用的.validate()函数是另一个易用的 Alamofire 特性。将其与请求和响应链接，以确认响应的状态码在默认可接受的范围（200到299）内。如果认证失败，响应处理方法将出现一个相关错误，我们可以根据不同在完成处理方法中处理这个错误。
比如下面的样例，成功时会打印成功信息，失败时输出具体错误信息。
Alamofire.request("https://httpbin.org/get", parameters: ["foo":"bar"])
.validate()
.responseJSON { responsein
switchresponse.result.isSuccess {
casetrue:
print("数据获取成功!")
casefalse:
print(response.result.error)
}