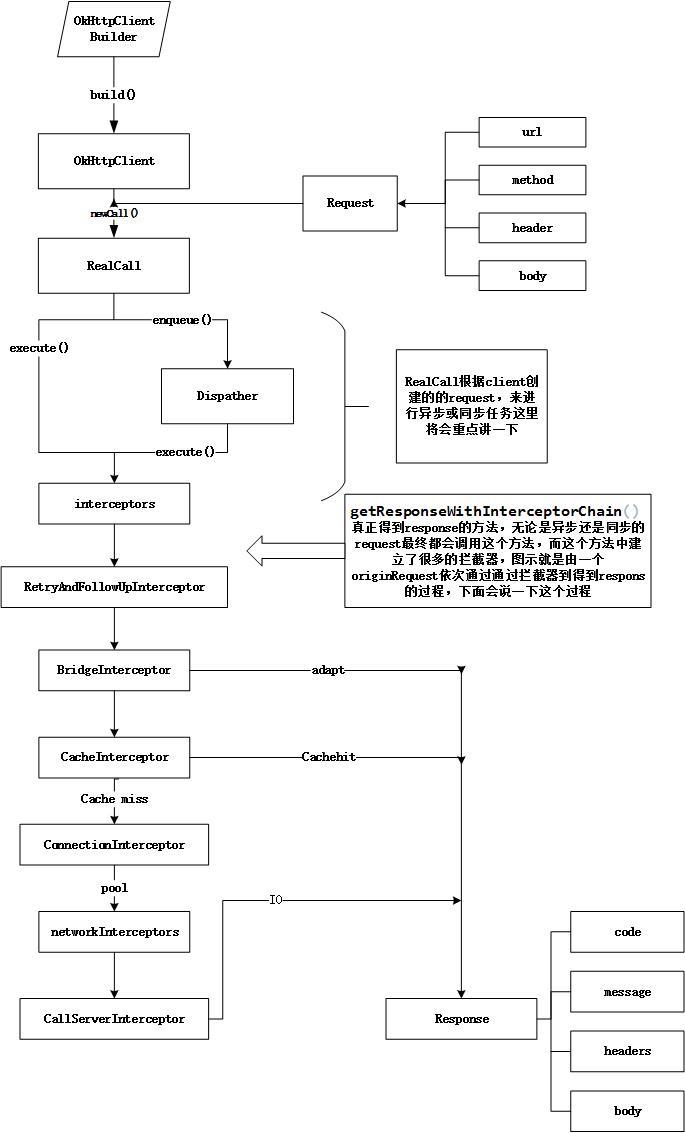
周报

学习并使用OkHttp3封装类：

OkHttp3是目前非常常用的通信类，整个流程是，通过OkHttpClient将构建的Request转换为Call，然后在RealCall中进行异步或同步任务，最后通过一些的拦截器interceptor发出网络请求和得到返回的response。



从上面的图可以看出，无论是同步请求还是异步请求，整个流程都是一样的，以下以GET方式来演示下同步与异步的区别：

**同步：**

public String get**(**String url**)** **throws** IOException **{**

//新建OKHttpClient客户端

OkHttpClient client **=** **new** OkHttpClient**();**

//新建一个Request对象

Request request **=** **new** Request**.**Builder**()**

**.**url**(**url**)**

**.**build**();**

//Response为OKHttp中的响应

Response response **=** client**.**newCall**(**request**).**execute**();**

**if** **(**response**.**isSuccessful**())** **{**

**return** response**.**body**().**string**();**

**}else{**

**throw** **new** IOException**(**"Unexpected code " **+** response**);**

**}**

**}**

**异步：**

private final OkHttpClient client **=** **new** OkHttpClient**();**

public void run**()** **throws** Exception **{**

Request request **=** **new** Request**.**Builder**()**

**.**url**(**"http://publicobject.com/helloworld.txt"**)**

**.**build**();**

client**.**newCall**(**request**).**enqueue**(new** Callback**()** **{**

@Override

public void onFailure**(**Call call**,** IOException e**)** **{**

e**.**printStackTrace**();**

**}**

@Override

public void onResponse**(**Call call**,** Response response**)** **throws** IOException **{**

**if** **(!**response**.**isSuccessful**())** **throw** **new** IOException**(**"Unexpected code " **+** response**);**

Headers responseHeaders **=** response**.**headers**();**

**for** **(**int i **=** 0**,** size **=** responseHeaders**.**size**();** i **<** size**;** i**++)** **{**

System**.**out**.**println**(**responseHeaders**.**name**(**i**)** **+** ": " **+** responseHeaders**.**value**(**i**));**

**}**

System**.**out**.**println**(**response**.**body**().**string**());**

**}**

**});**

**}**

总结：

拦截器采用了责任链设计模式,它将请求一层一层向下传，知道有一层能够得到Resposne就停止向下传递，然后将response向上面的拦截器传递，然后各个拦截器会对respone进行一些处理，最后会传到RealCall类中通过execute来得到esponse。