

上面是OKHttp总体设计图，主要是通过Diapatcher不断从RequestQueue中取出请求（Call），根据是否已缓存调用Cache或Network这两类数据获取接口之一，从内存缓存或是服务器取得请求的数据。该引擎有同步和异步请求，同步请求通过Call.execute()直接返回当前的Response，而异步请求会把当前的请求Call.enqueue添加（AsyncCall）到请求队列中，并通过回调（Callback）的方式来获取最后结果。

一、okhttp类使用建造者模式来返回OkHttpClient 的对象（设置缓存，连接时间，代理等），如果要对okhttpclient进行配置，则使用new出来的对象进行配置

二、Request类通过builder去设置url,get/post/body/header等

三、通过okhttpclient.newCall()去new出一个Call的实现类RealCall，然后在去执行enqueue，

这个方法中使用okhttpclent的dispatche（调度）方法去获得dispatcher的对象，再去调用dispatcher的enqueue()方法

四、dispatcher中的enqueue方法先判断异步队列中的请求数量是否小于最大请求数，如果小于，就将请求保存到正在运行的请求队列，相反就保存在准备的队列中。

五、enquene方法中需要new 一个AsyncCall异步任务类传入构造方法,AsynaCall的父类实现了Runnable接口，在run()里执行execute()。

六、getResponseWithInterfaceChain()使用责任链模式调用拦截器

1. 第一步

创建OkhttpClient对象和Request对象，并使用builder设计模式进行配置数据

1. 第二步

获取call对象，执行通过操作或者异步操作

1. 第三步

RealCall.enqueue()方法获取dispatcher对象，dispatcher再去执行dispatcher.enqueue方法，将请求放到请求队列中。并且创建线程池。然后执行线程池

1. 第四步

执行线程池的running方法，在该方法中添加所有执行拦截器，将拦截器（请求头/响应头拦截器，缓存拦截器，连接拦截器）依次执行，执行当前拦截器的时候同时执行下一个拦截器，而当前拦截器的response,依靠下一个response,直到最后一个拦截器执行完，再依次返回，得到最终的response,在将response回调返回