Retrofit 2.3.0源码解析

起源：okhttp存在的问题

- 请求较为抽象，除了必要的method,header,url，其它重要信息由RequestBody提供，这个类相对抽象，不易构建
  - 方案：采用Restful风格，接口的方法即代表一个请求，所需信息皆由注解-反射进行收集，简单明了，见方法即可明白该请求的用法
- 响应较为抽象，除了必要的header，protocal,code,statusmsg等信息，响应体由ResponseBody这个类提供，客户端还需进一步处理
  - 方案：通过在请求构建阶段就引入目标类型的泛型，并增加插拔式的转化器Converter，来进行抽象结果ResponseBody向目标响应对象的转换，解耦，灵活
- 请求--响应，发生在一个线程或者子线程里，对安卓等平台不友好
  - 方案：通过CallAdapter和Platform，对Call<T>进行装饰或者适配，使得在执行前后可以增加功能，甚至转成别的形式，适配安卓平台，包括Rxjava的使用

注解

- 请求方法注解：除了@HTTP之外，其它的注解都代表了一个Http方法，都接收一个 相对或者绝对url为值，当方法第一个参数被@URL注解时，这个值是可选项
  - @GET
  - @POST
  - @PUT
  - @DELETE
  - @HEAD
  - @PATCH
  - @OPTIONS
  - @HTTP eg.@HTTP(method = "GET", path = "blog/{id}", hasBody = false)
- 媒体类型注解
  - @FormUrlEncoded：说明请求将会使用表单形式，对应请求Content-Type为application/x-www-form-urlencoded，参数应使用Field注解，将会被utf-8进行encode
  - @Multipart:表明该请求是multi-part，参数应该带有Part对象，参数应该使用@Part注解
  - @Streaming:其它情况下，响应数据都是一次性载入内存。使用该注解，则不用将响应体的流一次性载入内存，适用于大数据量/文件传输。BuiltInConverters里判断，需要目标响应类为ResponseBody
- 请求参数注解
  - @Headers：作用于方法，静态添加请求头
  - @Header。@HeaderMap：作用于参数，动态的添加请求头
  - @Path: 作用于URL，替换{代码片段}，用以组合成最终url
  - @Query,@QueryMap：作用于参数，一般作为Url后的？的参数，会被encoded
  - @Field,@FieldMap：作用于参数，一般作为POST表单请求的参数，会被encoded
  - @Part：表示multi-part模式下的单个部分，一般用来上传文件，@Part注解的类型可分三类：1.Multipart.part，则注解的value不需要值，因为这已经是个完整的part;2.RequestBody，则需要提供part的name（如有表单上传文件，则还需要提供file-name:..具体见multi-part格式），使用是根据其定义的Content-type来进行；3.其它情况，则需要Converter参与转换
  - @PartMap:代表了一个multi-part请求中的键值对部分；该注解下的键值对中的值会有两种情况，1、RequestBody,它将会根据其content-type进行使用；2、其它类型，需要Converter参与
  - @Url:该注解与baseurl互斥，该注解的值代表了一个完整的url地址
  - @Body，作用于方法参数，直接控制POST/PUT方法的请求体，而不是作为请求参数或者表单形式的请求，该对象需要使用Converter来进行序列转化，转化结果将作为请求体直接发送

流程

1. 创建Retrofit对象，指定Call工厂，CallAdapter工厂，Converter工厂
2. create 定义Service接口，里面的方法即能代表一个http请求
3. 为Service创建了动态代理，使得其定义的方法在调用时，都会走InvocationHandler的invoke方法
4. invoke方法会返回一个Call<T>对象（可通过CallAdapter转换），默认为OkHttpCall，这个对象类型与接口方法声明的返回值类型要一致。
5. invoke方法中，通过调用loadServiceMethod，将方法转为ServiceMethod这个类
6. 调用Call对象的execute()或者enqueue方法，开启一次连接，其内部实现都是创建了一个okhttp内部的call来进行，这个Call对象相当于一个装饰器。创建方法是createRawCall()。
7. createRawCall()，内部是根据servicemethod和方法的参数列表来进行的。这个ServiceMethod是在动态代理执行过程中由接口方法转换而来。
8. 创建了okhttp的raw-call之后，进行基于okhttp的一次网络通信，得到相应结果。
9. 得到响应结果后，通过方法parseResponse(okhttp3.Response rawResponse)来进行结果解析，对于不成功的情况（http-code不是200），不用关注响应体，返回对应结果。对于请求成功（code=200）的，需要通过ResponseBody->T 的Converter进行结果转换。得到最终结果之后，同步请求则直接返回结果，异步请求则通过CallBack回调结果。
10. 注意：对于安卓系统，通过CallAdapter转换后的Call是ExecutorCallbackCall，它相当于是个装饰器，包装了OkHttpCall这个对象，额外的进行了线程切换等操作

CallAdapter：
将一个Call(带有返回值类型R，泛型)转换为类型T，具体子类由工厂类Factory创建，工厂模式，工厂在Retrofit创建时添加。
该类主要目的是：为了给客户端提供更加灵活的请求执行方式，比如安卓平台的结果回调需要在主线程，Rxjava的事件流调用。注意，无论如何适配，底层最终都由okhttp进行网络通信，所有适配后的新类型T，里面仍要执行Call.enqueue等方法。
Type responseType(); 当将一个http响应转换为目标对象时，需要的返回值类型，
e.g.Call<Repo> 中的Repo才是。
T adapt(Call<R> call);将一个请求类型Call的请求转化为T类型。该方法目前仅在动态代理返回值处使用。

Converter：
将一个对象与HTTP的表示进行双向转换，其实例由 抽象工厂模式创建，工厂是在Retrofit创建时添加。
Factory：
抽象工厂，负责一系列Converter的创建，主要是3种类型的Converter:
T->RequestBody的Converter:主要用于@Part,@Body,@PartMap注解的参数转为请求；
T->String的Converter: 主要用于将@Query,@Filed，@Header等参数注解转为string参数；
ResponseBody ->T的Converter，主要用于将响应体转为目标对象。

Retrofit：
外观类，负责与客户端交互，并维护一些配置选项。建造者模式构建，以便链式调用+已构建的对象不可变；
主要配置项：baseurl,call.factory.calladapter.factory,converter.factorty,callback-executor,platform;
核心方法：
<T> T create(final Class<T> service)：为接口创建动态代理，在接口方法调用时，可将java方法转为servicemethod,进一步可翻译成一个http请求。并使用方法对应的adapter将Call<T>转为目标请求格式（转换后的请求格式，底层仍要通过okhttp的call进行请求。）

loadServiceMethod(Method method)：将java方法转为servicemethod，并缓存；

nextCallAdapter(.....)：根据返回值类型，提供一个CallAdapter，以适配该方法的请求格式。

nextRequestBodyConverter(.....)：返回一个Converter，用来将方法参数转为RequestBody；

nextResponseBodyConverter(.....): 返回一个Converter，用来将ResponseBody转为对应目标结果类型；

Retrofit build()：构建Retrofit对象，其中，Call.Factory默认为OkHttpClient，回调处理器默认为平台默认的处理器，对于安卓平台，则为MainThreadExecutor；Calladapter.Factory列表里，默认添加平台下的适配器工厂，参数为回调处理器；
Converter.Factory列表，默认添加了BuiltInConverters；

Platform:
平台类，用来根据使用平台差异，提供不同的处理方法。主要方法：
findPlatform：通过反射平台核心类来获取对应平台；
defaultCallbackExecutor：默认的回调处理器，安卓里是MainThreadExecutor，里面通过主线程的Handler实现请求结果的线程切换；
defaultCallAdapterFactory，默认的请求适配器工厂，安卓里面，需要持有回调处理器的引用，用于切换线程。

动态代理：根据声明的接口A，类加载器等信息，动态的生成一个类，它继承于Proxy（构造函数里有InvokeHandler参数，以便方法代理使用），并实现了A。然后通过反射+构造函数(含有Invokdhandler..)获取该类的一个对象，当调用A声明的方法时，会转给invokehandler的invoke方法处理

ServiceMethod：
将一个接口方法的调用转换为一次HTTP的请求。建造者模式构建，所需要的核心参数：
Retrofit提供的配置项，Mehod本身，还有method的各种注解。
核心方法：
toRequest(@Nullable Object... args)：通过方法的各项参数，来构建一个HTTP请求；主要涉及RequestBuilder和ParameterHandler两个类；
toResponse(ResponseBody body)：将ResponseBody 转换为目标结果形式，主要是通过Converter进行转换。

ServiceMethod对象由建造者模式创建，build()方法是该类的核心，具体流程：
1.根据接口方法的返回值+方法上的注解，从注册到Retrofit的CallAdapter的工厂里找到对应的工厂，并创建一个CallAdapter;
2.根据CallAdapter里的返回值类型（方法返回值的参数化类型）+方法上的注解，从注册到Retrofit的Converter的工厂列表里找到对应的工厂，并为该方法创建一个ResponseBody->T 类型的Converter。
3.开始遍历解析方法注解，主要是http请求方法，路径path，静态header以及请求类型（e.g multi-part),是否有body。
4.获得接口方法参数的注解列表，依次解析，根据不同的注解类型，获得不同泛型的ParameterHandler对象，每种对象持有一个T->RequestBody 或者 T->String 类型的Converter。以在后续创建请求时，将不同的注解参数转换为对应的http请求参数。
5.生成ServiceMethod对象。

ParameterHandler<T>:
将类型T的值，转为对应的RequestBody或者String，然后添加到RequestBuilder中，每一个该对象，都持有一个Converter对象。

Call<T>：
代表了Retrofit方法的一次调用，调用过程中会向服务器发送一个请求，并获得响应结果。本质上来说，这个接口是对Okhttp中Call接口的泛型封装，方法基本同Call，以便处理请求和获得目标响应。核心方法：
Response<T> execute()：同步执行请求，响应结果带有T类型的目标对象。
void enqueue(Callback<T> callback);异步执行请求，回调的响应结果带有T类型的目标对象。
Request request();:获得原始的okhttp Request。
该接口有2个子类：
OkHttpCall；
ExecutorCallbackCall

Response<T>,
CallBack<T>
主要是为了响应结果能转换为目标结果格式存在的，与okhttp中的同名类使用类似。

RequestBuilder：
根据之前 注解-反射-转换得到的url,method.参数，请求头，媒体类型，body等信息，构建Request

OkHttpCall：
Call<T>的实现类，持有 okhttp3.Call对象，持有ServiceMethod对象，持有接口方法的参数列表。
在createRawCall（）方法中，通过ServiceMethod对象创建了okhttp的Request，进一步由OkHttpClient生成了okhttp3.Call；
请求执行方法execute()和enqueue（）底层都是okhttp3.Call实现的。
对于响应结果，需要结合Converter来进行处理，若是异步请求，则通过Callback<T>对象进行回调。

ExecutorCallbackCall：
装饰器类，内部实际包装了OkHttpCall，增加的功能主要是将响应结果切换到主线程