# Retrofit 翻译

## Request method 请求方法

所有请求方法必须 有一个HTTP 注解 提供请求方法和相关 的URL 。 retrofit 提供了5中注解 代表了不同的方法 （GET, POST, PUT, DELETE, and HEAD） 相关的资源url 在注解中会被明确：

@GET("users/list")

也可以在url中明确查询参数

@GET("users/list?sort=desc")

## URL MANIPULATION url 操纵 修改

一个请求的url 可以动态的使用 替换块和参数来更新。 一个替换块 就是一个使用“{”“}” 包裹的文字表达式。 一个相对的参数必须被 @Path使用相同的名字注解

Demo：

@GET("group/{id}/users")

Call<List<User>> groupList(@Path("id") int groupId);

也可以通过注解添加查询参数

@GET("group/{id}/users")

Call<List<User>> groupList(@Path("id") int groupId, @Query("sort") String sort);

For complex query parameter combinations a Map can be used.

对于复杂的查询 参数组合 可以使用一个Map

@GET("group/{id}/users")

Call<List<User>> groupList(@Path("id") int groupId, @QueryMap Map<String, String> options);

## REQUEST BODY

一个对象可以被作为一个http 请求的body 使用@Body 注解

@POST("users/new")

Call<User> createUser(@Body User user);

对象会同样被转换 使用明确在retrofit 实例上， 如果没有转换者被指定 只用RequestBody可以使用。

## FORM ENCODED AND MULTIPART 表单和组合请求

方法同时可以被声明来 发送表单数据以及多部分数据，

表单数据 会被发送 当使用@FormUrlEncoded 表示在方法上时。 所有的键值对 使用@Field 注解 表示名字和提供值得对象。

Demo：

@FormUrlEncoded

@POST("user/edit")

Call<User> updateUser(@Field("first\_name") String first, @Field("last\_name") String last);

组合请求会被使用当 方法被@Multipart 表示时 分块使用@Part 注解来表示。

@Multipart

@PUT("user/photo")

Call<User> updateUser(@Part("photo") RequestBody photo, @Part("description") RequestBody description);

组合请求块使用一个 retrofit 的 转换器或者 他们可以实现RequestBody 接口来处理他们自己的序列化。

## HEADER MANIPULATION 请求 头部修改

你可以使用@Headers 注解为方法设置静态的请求头

Demo:

@Headers("Cache-Control: max-age=640000")

@GET("widget/list")

Call<List<Widget>> widgetList();

@Headers({

"Accept: application/vnd.github.v3.full+json",

"User-Agent: Retrofit-Sample-App"

})

@GET("users/{username}")

Call<User> getUser(@Path("username") String username);

注意 头部们不会覆盖对方。 所有的头部 会被以相同的名字 包含在请求内。

一个请求头 可以使用@header 注解 动态更新。 一个相应参数必须被提供改@header

如果值是null 头部会被省略 。否则 值的toString方法会被调用把他的结果使用。

@GET("user")

Call<User> getUser(@Header("Authorization") String authorization)

所有请求都需要添加的头部可以使用OKHttp 拦截器来指定。

## SYNCHRONOUS VS. ASYNCHRONOUS 同步与异步。

Call 实例可以 同步的产生也可以异步的产生。 所有的实例只能被使用一次。 但是使用clone 可以创建一个新的完全相同的实例来使用。

在Android 上回调 会执行在主线程上 在JVM上 回调会发生在请求发起的线程上。

## Retrofit Configuration 配置

Retrofit 是 把你的api接口转化为callable对象的类。 默认情况下， retrofit 会给你智能的设置对应你的平台， 但是他运行自定义。

转换器：

默认情况下 ， retrofit 只能 反序列话 http body 在 okhttp 的responseBody 类型，并且智能接受RequestBody 类型作为@body的参数

转换器可以添加支持的类型

常用的转换器

* [Gson](https://github.com/google/gson): com.squareup.retrofit2:converter-gson
* [Jackson](http://wiki.fasterxml.com/JacksonHome): com.squareup.retrofit2:converter-jackson
* [Moshi](https://github.com/square/moshi/): com.squareup.retrofit2:converter-moshi
* [Protobuf](https://developers.google.com/protocol-buffers/): com.squareup.retrofit2:converter-protobuf
* [Wire](https://github.com/square/wire): com.squareup.retrofit2:converter-wire
* [Simple XML](http://simple.sourceforge.net/): com.squareup.retrofit2:converter-simplexml
* Scalars (primitives, boxed, and String): com.squareup.retrofit2:converter-scalars

Demo：

Retrofit retrofit = new Retrofit.Builder()

.baseUrl("https://api.github.com")

.addConverterFactory(GsonConverterFactory.create())

.build();

GitHubService service = retrofit.create(GitHubService.class);