# 什么是

相信大家对Retrofit一定不陌生，或多或少都听过它的名字，它是一个最近很流行也很好用的网络请求库，它对okHttp进行了封装，加上利用了Java注解等特性，使得它能适应于灵活多变的网络请求情景。那么接下来我们一起学习怎样使用Retrofit2。

# ****Retrofit使用方法****

　　由于retrofit2.0与先前版本的差别还是比较大，对于不同版本之间的差异在这里就不在进行详细区别。下面的例子也是针对于retrofit2.0进行介绍的。retrofit2.0它依赖于OkHttp,而且这部分也不再支持替换。在这里我们也不需要显示的导入okHttp,在retrofit中已经导入okhttp3。

<dependency>

<groupId>com.squareup.okhttp3</groupId>

<artifactId>mockwebserver</artifactId>

<scope>test</scope>

</dependency>

* 1
* 2
* 3
* 4
* 5

　　在下面的例子当中采用与GitHub一些相关api进行演示。在这里首先需要添加访问网络的权限。

<uses-permission android:name="android.permission.INTERNET"/>

* 1

## ****简单示例****

### ****添加Gradle依赖项****

　　在这里我们最好查看一下[retrofit的官网](http://square.github.io/retrofit/)添加最新依赖。

compile 'com.squareup.retrofit2:retrofit:2.0.1'

* 1

### ****创建API接口****

　　在retrofit中通过一个Java接口作为http请求的api接口。

public interface GitHubApi {

@GET("repos/{owner}/{repo}/contributors")

Call<ResponseBody> contributorsBySimpleGetCall(@Path("owner") String owner, @Path("repo") String repo);

}

* 1
* 2
* 3
* 4
* 5

### ****创建retrofit实例****

　　在这里baseUrl是在创建retrofit实力的时候定义的，我们也可以在API接口中定义完整的url。在这里建议在创建baseUrl中以”/”结尾，在API中不以”/”开头和结尾。

Retrofit retrofit = new Retrofit.Builder()

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

.build();

* 1
* 2
* 3

### ****调用API接口****

　　在调用API接口请求后，获得一个json字符串，通过Gson进行解析，获得login以及contributions。

GitHubApi repo = retrofit.create(GitHubApi.class);

Call<ResponseBody> call = repo.contributorsBySimpleGetCall(mUserName, mRepo);

call.enqueue(new Callback<ResponseBody>() {

@Override

public void onResponse(Call<ResponseBody> call, Response<ResponseBody> response) {

try {

Gson gson = new Gson();

ArrayList<Contributor> contributorsList = gson.fromJson(response.body().string(), new TypeToken<List<Contributor>>(){}.getType());

for (Contributor contributor : contributorsList){

Log.d("login",contributor.getLogin());

Log.d("contributions",contributor.getContributions()+"");

}

} catch (IOException e) {

e.printStackTrace();

}

}

@Override

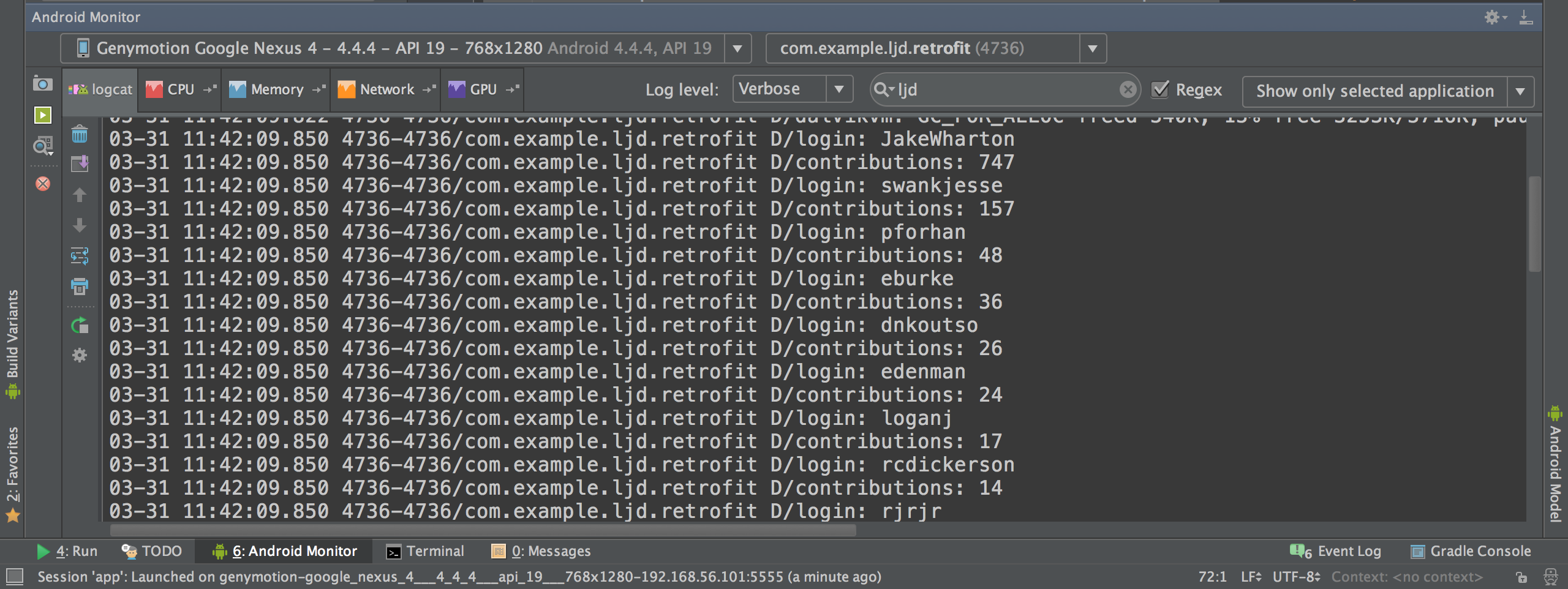
public void onFailure(Call<ResponseBody> call, Throwable t) {

}

});

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23

### ****效果展示****

　　这样就完成了一个http请求，上面请求的完整地址为：<https://api.github.com/repos/square/retrofit/contributors>   
　　然后我们看一下运行结果:   


## ****取消请求****

　　我们可以终止一个请求。终止操作是对底层的httpclient执行cancel操作。即使是正在执行的请求，也能够立即终止。

call.cancel();

* 1

# 为什么要用

# 优点

# 缺点