AndroidHint Lv2
2019年09月13日 阅读 3158

关注

OkHttp与Retrofit的作用和联系

一、OkHttp

1、OkHttp的介绍

OkHttp是一个关于网络请求的第三方类库，其中封装了网络请求的get、post等操作的底层实现，是Android端目前最为火热的网络请求框架之一。

2、OkHttp的使用

在Android Studio中不需要下载专门的jar包，直接在gradle中添加依赖，如下所示：

复制代码

```
compile 'com.squareup.okhttp3:okhttp:3.10.0'
```

3、get方法请求

- 同步请求 同步请求需要在子线程中执行，并且在执行结果返回后在UI线程中修改UI。下面是一个get请求的代码例子：

复制代码

```
new Thread(new Runnable() {
    @Override
    public void run() {
        OkHttpClient client = new OkHttpClient();
        Request request = new Request.Builder().url("http:www.taobao.com").build();
        try {
            Response response = client.newCall(request).execute();
            if (response.isSuccessful()) {
```

```
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}).start();
```

get同步请求分为三步： 首先使用new创建OkHttpClient对象。 然后调用OkHttpClient对象的newCall方法生成一个Call对象，该方法接收一个Request对象，Request对象存储的就是我们的请求URL和请求参数。 最后执行Call对象的execute方法，得到Response对象，这个Response对象就是返回的结果。

- 异步请求 异步方法不需要开启一个线程执行操作，但是要注意的是最终的回调方法是在子线程执行的，所以我们要修改UI也必须要在UI线程中进行修改。看一下下面这个例子：

复制代码

```
OkHttpClient client = new OkHttpClient();
Request request = new Request.Builder().url("http://www.tabao.com").build();
client.newCall(request).enqueue(new Callback() {
    @Override
    public void onFailure(Call call, IOException e) {
        Log.d("ABC", "error");
    }

    @Override
    public void onResponse(Call call, Response response) throws IOException {
        if (response.isSuccessful()) {
            Log.d("ABC", "response=" + response.body().toString());
            // 转到UI线程去修改UI
        }
    }
});
```

get异步请求分为三步： 首先使用new创建OkHttpClient对象。 然后调用OkHttpClient对象的newCall方法生成一个Call对象，该方法接收一个Request对象，Request对象存储的是我们的请求URL和请求参数。 最后执行Call对象的enqueue方法，该方法接收一个Callback回调对象，在回调对象的onResponse方法中拿到Response对象，这就是返回的结果。

总结：get的同步方法和异步方法的差别在于同步方法需要手动开启一个线程执行，而异步方法不需要（其实是使用了内部的线程）。

4、post方法请求

- 同步请求 post方法的同步请求和get方法的同步请求几乎是一样的，看下面的例子，并和上面的get的同步请求进行对比：

复制代码

```
new Thread(new Runnable() {
    @Override
    public void run() {
        OkHttpClient client = new OkHttpClient();
        FormBody.Builder formBody = new FormBody.Builder();
        formBody.add("bookName", "Android Art");
        Request request = new Request.Builder().url("http:www.taobao.com").post(formBody.bu
        try {
            Response response = client.newCall(request).execute();
            if (response.isSuccessful()) {
                Log.d("ABC", "response=" + response.body().toString());
                // 转到UI线程去修改UI
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}).start();
```

对比：post方法的同步请求和get方法的同步请求的区别在于，post方法生成Request对象时多执行了post(RequestBody)方法，而RequestBody对象的子类是FormBody类，所以可以使用FormBody对象创建键值对参数。

再深入一步，为什么post请求需要执行post方法，而且get请求不用呢？我们看一下post方法里面执行了什么操作？

复制代码

```
public Builder post(RequestBody body) {
    return method("POST", body);
}
```

返回了method方法的执行结果，这里传递了一个“POST”字符串，看一下method方法的执行：

复制代码

```
public Builder method(String method, RequestBody body) {
  if (method == null) throw new NullPointerException("method == null");
  if (method.length() == 0) throw new IllegalArgumentException("method.length() == 0");
  if (body != null && !HttpMethod.permitsRequestBody(method)) {
    throw new IllegalArgumentException("method " + method + " must not have a request
  }
  if (body == null && HttpMethod.requiresRequestBody(method)) {
```

```
    this.method = method;
    this.body = body;
    return this;
}
```

这里将“POST"字符串传递给了Builder对象的method变量，而Builder的默认构造函数中将method变量赋值为"GET"。到这里就水落石出了，原来Request.Builder对象创建时，默认就是get请求，所以get请求就不用设置get方法了，而post请求则需要修改method变量的值为"POST"。

- 异步请求 post方法的异步请求和get方法的异步请求也是非常相似的，区别也是同步请求的区别，所以直接看例子吧：

复制代码

```
OkHttpClient client = new OkHttpClient();
FormBody.Builder formBody = new FormBody.Builder();
formBody.add("bookName", "Android Art");
Request request = new Request.Builder().url("http://www.tabao.com").post(formBody.build()).b
client.newCall(request).enqueue(new Callback() {
    @Override
    public void onFailure(Call call, IOException e) {
        Log.d("ABC", "error");
    }

    @Override
    public void onResponse(Call call, Response response) throws IOException {
        if (response.isSuccessful()) {
            Log.d("ABC", "response=" + response.body().toString());
            // 转到UI线程去修改UI
        }
    }
});
```

二、Retrofit

1、Retrofit的介绍

Retrofit是一个RESTful的HTTP网络请求框架，它是基于OkHttp的。它是通过注解配置网络参数的，支持多种数据的解析和序列化（Gson、Json、Xml等，并且对RxJava也是支持的。下面是Retrofit和OkHttp的联系图（图来自于https://blog.csdn.net/carson_ho/article/details/73732076）：

可以看到App应用层是通过Retrofit请求网络，然后使用Retrofit接口层封装请求参数、Header、URL等信息，然后由OkHttp完成后续的和服务器的交互。 而服务器返回响应数据后，也是先返回给OkHttp，然后OkHttp将原始结果返回给Retrofit，Retrofit根据用户的需求对原始数据进行解析封装，并返回给App应用层。

2、Retrofit的使用

2.1 添加依赖库

在Android Studio中Retrofit库的依赖，由于其还依赖了OkHttp库，所以同时需要添加OkHttp的依赖。

复制代码

```
compile 'com.squareup.retrofit2:retrofit:2.1.0'
compile 'com.squareup.okhttp3:okhttp:3.8.1'
```

2.2 添加网络请求的接口

Retrofit使用**注解**的方式描述和配置网络请求参数。实际上是运用动态代理的方式将注解翻译成一个Http请求，然后执行该请求。基本的结构如下：

复制代码

```
public interface GitHubService {
  @GET("getUserData")
  Call<User> getCall();
}
```

User.java

复制代码

```
public class User {
    private int userId;
    private String userName;
    private ExInfo exInfo;

    private static class ExInfo {
        private String from;
        private String birthDay;
    }

    public void show() {
        Log.d("ABC", "userId=" + userId);
```

```
            Log.d("ABC", "birthday=" + exInfo.birthDay);
        }
    }
```

注意项： 1、这是一个接口（Interface）不是一个类（class）。 2、其中可以创建不同的接口方法。 3、每一个接口方法和接口方法的参数必须使用注解的方式标注，否则会报错。 4、请求方法有GET、POST、PUT、HEAD、DELETE等。 5、方法的返回类型必须为Call，xxx是接收数据的类（可以是我们自定义的类，也可以是ResponseBody）。 6、User中字段的名字必须和后台Json定义的字段名是一致的，这样Json才能解析成功。

2.3 创建Retrofit对象

复制代码

```
Retrofit retrofit = new Retrofit.Builder().baseUrl("http://10.75.114.138:8081/")
                .addConverterFactory(GsonConverterFactory.create()).build(); //设置网络请求的U
                .addConverterFactory(GsonConverterFactory.create()) //设置数据解析器
                .build();
```

网络请求的地址：创建Retrofit对象时通过baseUrl()设置+网络请求接口的注解设置。 addConverterFactory(GsonConverterFactory.create())的作用是设置数据解析器，它可以解析Gson、JsonObject、JsonArray这三种格式的数据。

2.4 创建网络请求接口实例

复制代码

```
// 创建 网络请求接口 的实例
GitHubServiceRequest request = retrofit.create(GitHubService.class);

//对 发送请求 进行封装
Call<Reception> call = request.getCall();
```

2.5 发送网络请求

同步请求：

复制代码

```
// 发送网络请求（同步）
Response<Reception> response = call.execute();
```

异步请求：

复制代码

```
call.enqueue(new Callback<User>() {
    //请求成功后回调
    @Override
    public void onResponse(Call<User> call, Response<User> response) {
        //处理请求结果
        User user = response.body();
        if(user != null) {
            user.show();
        }
    }

    //请求失败后回调
    @Override
    public void onFailure(Call<User> call, Throwable throwable) {
        System.out.println("请求失败");
    }
});
```

3、总结

Retrofit的一个RESTful风格的网络请求框架，其下一层的实现也是OkHttp，所以其原理和OkHttp的一样的，只是在OkHttp的上面封装了一层，使请求接口和数据解析更加简洁明了。

文章分类 Android 文章标签 面试

小只前端攻城狮 1天前 面试 前端

🔥EventLoop是什么？执行机制是什么？

前言 什么是 EventLoop？ 先看这张图，先不管宏任务，微任务是什么，先看整个流程。 分析： 判断宏任务队列是...

501 14 1

敖丙 6天前 Java 面试

模板方法 （宝，我输液了，输的想你的夜）

大家每到一家公司都会发现，每个公司都会有一个规范，比如说请假流程规范，代码规范等...

8218 66 21

沉默王二 18小时前 后端 面试

被读者告白了！

见下图，微信上收到读者发来了一篇有道云笔记《致前辈》。 我以为最多也就 200 个字，结果打开一看，惊呆了，...

264 2 评论

神奇的程序员 1月前 程序员 面试

我离职了

大家看到这篇文章的时候，我已经从老东家离职了，从有离职想法到找到新工作，期间经历...

4.9w 972 755

敖丙 7天前 程序员 面试

毕业十年后，他年薪百万，我年薪刚破十万，人和人之间的差距怎么这么...

又是一年毕业季，翻看朋友圈，当年的同学现如今都已买车买房、结婚生子，而自己还在担...

3992 22 18

魔王哪吒 23天前 JavaScript 面试

一个合格的中级前端工程师需要掌握的技能笔记（上）

每天学习编程，让你离梦想更新一步，感谢不负每一份热爱编程的程序员，不论知识点多么...

1.4w 390 16

程序员十三 1月前 Vue.js 面试

Vue3发布半年我不学，摸鱼爽歪歪，哎~就是玩儿

是从 Vue 2 开始学基础还是直接学 Vue 3？ 尤雨溪给出的答案是：“直接学 Vue 3 就行了，...

4.1w 1346 176

聊聊前端面试

最近 Zoom 国内又开放招聘了，我们组有了前端的 HC，所以我也参加了几场面试。在面试...

1.9w 470 71

敖丙 13天前 Java 面试

如何用策略模式，优化你代码里的的if-else?

最近有一个学妹在跟我沟通如何有效的去避免代码中一长串的if else判断或者switch条件判...

7249 76 27

图雀社区 3月前 JavaScript 面试

字节跳动最爱考的前端面试题：JavaScript 基础

最大安全数字：Number.MAX_SAFE_INTEGER = Math.pow(2, 53) - 1，转换成整数就是 1...

7.9w 2911 193
