# OkHttpUtils

OkHttpUtils - 封装了okhttp的网络框架，目前Get，Post的请求已经完成，支持大文件上传下载，上传进度回调，下载进度回调，表单上传（多文件和多参数一起上传），链式调用，整合Gson，自动解析返回对象，支持Https和自签名证书，支持cookie自动管理，后期将要实现的功能，统一的上传管理和下载管理。该项目是根据：<https://github.com/hongyangAndroid/okhttp-utils> 和 <https://github.com/pengjianbo/OkHttpFinal> 修改而成，喜欢原作的可以去使用。同时欢迎大家下载体验本项目，如果使用过程中遇到什么问题，欢迎反馈。

## 用法

* Android Studio

使用前，对于Android Studio的用户，可以选择添加:

compile project(':okhttputils')

* Eclipse

自行copy源码。

****注意****

使用的okhttp的版本是最新的3.0版本，和以前的2.x的版本可能会存在冲突，整合了Gson，提供了自定Callback，可以按照泛型，自行解析返回结果：

compile 'com.android.support:support-annotations:23.1.1'

compile 'com.squareup.okhttp3:okhttp:3.0.0-RC1'

compile 'com.google.code.gson:gson:2.5'

## 目前支持

* 一般的get请求
* 一般的post请求
* 基于Http Post的文件上传（类似表单）
* 多文件和多参数同时上传
* 大文件下载和下载进度回调
* 大文件上传和上传进度回调
* 支持session的保持
* 支持自签名网站https的访问，提供方法设置下证书就行
* 支持根据Tag取消请求
* 支持自定义泛型Callback，自动根据泛型返回对象

## 即将实现

* 统一的文件上传管理
* 统一的文件下载管理
* 采用线程池或者volley对普通请求进行管理

## 用法示例

### 1.普通的GET请求，根据泛型Bean返回值也是Bean

private void getJson() {

OkHttpUtils.get("http://192.168.1.111:8080/UploadServer/ResponseJson")//

.tag(this)//

.params("ppppppp", "ppp")//

.headers("hhhhhhh", "hhh")//

.execute(new MyBeanCallBack<Bean>() {

@Override

public void onResponse(Bean bean) {

System.out.println("onResponse:" + bean);

}

});

}

### 2.普通的POST请求，根据泛型Bean返回值也是Bean

private void responseJson() {

OkHttpUtils.post("http://192.168.1.111:8080/UploadServer/ResponseJson")//

.tag(this)//

.params("ppppppp", "ppp")//

.headers("hhhhhhh", "hhh")//

.execute(new MyBeanCallBack<Bean>() {

@Override

public void onResponse(Bean bean) {

System.out.println("onResponse:" + bean);

}

});

}

### 3.普通Post，直接上传String类型的文本

不建议这么用，该方法上传字符串会清空实体中其他所有的参数，但头信息不清除，例如本例中的 params 参数不会上传

private void postString() {

OkHttpUtils.post("http://192.168.1.111:8080/UploadServer/UploadString")//

.tag(this)//

.params("ppppppp", "ppp")//

.headers("hhhhhhh", "hhh")//

.content("asdfasdfad这是文本这是文本aasfesr")//

.mediaType(PostRequest.MEDIA\_TYPE\_PLAIN)//

.execute(new MyBeanCallBack<String>() {

@Override

public void onResponse(String s) {

System.out.println("onResponse:" + s);

}

});

}

如果要上传Json，把上面的 mediaType 改为 .mediaType(PostRequest.MEDIA\_TYPE\_JSON)

### 4.表单Post，同时上传多文件和多参数（推荐使用）

private void uploadFile() {

OkHttpUtils.post("http://192.168.1.111:8080/UploadServer/UploadFile")//

.tag(this)//

.headers("aaa", "111")

.headers("bbb", "222")

.params("ccc", "333")

.params("ddd", "444")

.params("file1", new File(Environment.getExternalStorageDirectory() + "/DCIM/Camera/IMG\_20151225\_155549.jpg"))//

.params("file2", new File(Environment.getExternalStorageDirectory() + "/DCIM/Camera/IMG\_20160109\_010308.jpg"))//

.params("file3", new File(Environment.getExternalStorageDirectory() + "/video/splash.avi"))//

.execute(new MyBeanCallBack<String>() {

@Override

public void onResponse(String s) {

System.out.println("onResponse:" + s);

}

});

}

其中文件的key，相当于表单中<input type="file" name="File1"/>的name属性。

### 5.下载文件，get和post都可以，这里使用post演示

OkHttpUtils.post("http://192.168.1.111:8080/UploadServer/DownloadFile")//

.tag(this)//

.params("ppppppp", "ppp")//

.headers("hhhhhhh", "hhh")//

.execute(new MyFileCallBack(Environment.getExternalStorageDirectory() + "/video", "bbb.avi") {

@Override

public void onResponse(File response) {

System.out.println("onResponse:" + response);

}

});

具体FileCallBack内部实现的下载进度监听，自行看代码

### 6.根据tag取消请求

目前对于支持的方法都添加了最后一个参数Object tag，取消则通过OkHttpUtils.cancel(tag)执行。

例如：在Activity中，当Activity销毁取消请求，可以在onDestory里面统一取消。

@Override

protected void onDestroy() {

super.onDestroy();

OkHttpUtils.getInstance().cancelTag(this);

}

### 7.自定义CallBack

目前内部提供的包含BeanCallBack, StringCallBack, FileCallBack, BitmapCallback，可以根据自己的需求去自定义Callback

#### 其中BeanCallBack使用比较多，它支持传递一个泛型，将返回的Response对象解析成需要的类型并且返回，目前支持：

* 一般的 JavaBean
* 字符串 String
* 集合泛型 List<Bean>
* 集合泛型 Map<Bean>

以下是实现代码

public abstract class BeanCallBack<T> extends AbsCallback<T> {

@Override

public T parseNetworkResponse(Response response) throws Exception {

Type type = this.getClass().getGenericSuperclass();

if (type instanceof ParameterizedType) {

//如果用户写了泛型，就会进入这里，否者不会执行

ParameterizedType parameterizedType = (ParameterizedType) type;

Type beanType = parameterizedType.getActualTypeArguments()[0];

if (beanType == String.class) {

//如果是String类型，直接返回字符串

return (T) response.body().string();

} else {

//如果是 Bean List Map ，则解析完后返回

return new Gson().fromJson(response.body().string(), beanType);

}

} else {

//如果没有写泛型，直接返回Response对象

return (T) response;

}

}

}

通过parseNetworkResponse回调的response进行解析，该方法运行在子线程，所以可以进行任何耗时操作

### 8.上传下载的进度显示

public abstract class AbsCallback<T> {

/\*\* Post执行上传过程中的进度回调，get请求不回调，UI线程 \*/

public void upProgress(long currentSize, long totalSize, float progress) {

}

/\*\* 执行下载过程中的进度回调，UI线程 \*/

public void downloadProgress(long currentSize, long totalSize, float progress) {

}

}

callback回调中有upProgress 和 downloadProgress 方法，直接复写即可，在 get 请求中， upProgress 方法不会执行。

### 9.同步的请求

Response response = OkHttpUtils.get("http://www.baidu.com")//

.tag(this)//

.headers("aaa", "111")//

.params("bbb", "222").execute();

execute方法不传入callback即为同步的请求，返回Response对象，需要自己解析

### 10.全局配置

可以在Application中，通过：

try {

OkHttpUtils.debug(true, "MyOkHttp");

OkHttpUtils.getInstance()//

.setConnectTimeout(OkHttpUtils.DEFAULT\_MILLISECONDS)//

.setReadTimeOut(OkHttpUtils.DEFAULT\_MILLISECONDS)//

.setCertificates(new Buffer().writeUtf8(CER\_12306).inputStream())//

.setWriteTimeOut(OkHttpUtils.DEFAULT\_MILLISECONDS);

} catch (Exception e) {

e.printStackTrace();

}

然后调用 OkHttpUtils 的各种set方法。

### 11.为单个请求设置超时

比如涉及到文件的需要设置读写等待时间多一点。

private void responseJsonArray() {

OkHttpUtils.post("http://192.168.1.111:8080/UploadServer/ResponseJsonArray")//

.tag(this)//

.connTimeOut(2000)

.writeTimeOut(3000)

.readTimeOut(4000)

.params("ppppppp", "ppp")//

.headers("hhhhhhh", "hhh")//

.execute(new MyBeanCallBack<List<Bean>>() {

@Override

public void onResponse(List<Bean> beans) {

System.out.println("onResponse:" + beans);

}

});

}

### 12.自签名网站https的访问

非常简单，拿到xxx.cert的证书，然后调用

OkHttpUtils.getInstance().setCertificates(inputstream);

建议使用方式，例如我的证书放在assets目录：

try {

OkHttpUtils.getInstance()

.setCertificates(getAssets().open("srca.cer"), getAssets().open("aaa.cer"))//

} catch (Exception e) {

e.printStackTrace();

}