RxVolley使用文档 V1.1.0

RxVolley使用指南

RxVolley 项目地址： https://github.com/kymjs/RxVolley

概述

RxVolley是一个基于Volley的网络请求库；

同时支持RxJava；

可以选择使用OKHttp替代默认的 HttpUrlConnection 做网络请求；

可以选择使用图片加载功能(复用的网络请求将有效减少apk体积)；

移除了原Volley的 HttpClient 相关 API ，可在 API23 环境编译；

内置了RxBus的实现，可有效替换掉EventBus等相关库；

依赖

使用RxVolley，需要在你的build.gradle文件中加入

compile 'com.kymjs.rxvolley:rxvolley:1.1.0'

如果你还想使用OKhttp来替代默认的HttpUrlconnection，需要加入

compile 'com.kymjs.rxvolley:okhttp:1.1.0'

如果你想使用RxVolley的图片加载功能(复用http模块可以有效减少apk大小)，需要加入

compile 'com.kymjs.rxvolley:bitmapcore:1.1.0'

使用 RxVolley 做网络请求

简洁实现

//get请求简洁版实现  
RxVolley.get("http://www.kymjs.com/feed.xml", new HttpCallback() {  
 @Override  
 public void onSuccess(String t) {  
 Loger.debug("请求到的数据:" + t);  
 }  
});  
  
//post请求简洁版实现  
HttpParams params = new HttpParams();  
params.put("name", "kymjs");  
params.put("age", 18);  
params.put("image", new File("path"))//文件上传  
  
RxVolley.post("http://kymjs.com/feed.xml", params, new HttpCallback() {  
 @Override  
 public void onSuccess(String t) {  
 Loger.debug("请求到的数据:" + t);  
 }  
});

对Cookie等请求头的处理

//用户登录逻辑(HttpCallback中有很多重载方法，可以选择需要的实现)  
HttpParams params = new HttpParams();  
params.put("name", "kymjs");  
params.put("age", 18);  
params.put("password", "helloword");  
RxVolley.post("http://kymjs.com/login", params, new HttpCallback() {  
 @Override  
 public void onSuccess(Map<String, String> headers, byte[] t) {  
 Loger.debug("请求到的数据:" + new String(t));  
 // 获取到的cookie  
 Loger.debug("===" + headers.get("Set-Cookie"));  
 }  
});

//向服务器传递cookie信息  
HttpParams params = new HttpParams();  
params.put("name", "kymjs");  
params.put("age", 100);  
  
params.putHeaders("cookie", "your cookie");  
  
RxVolley.post("http://kymjs.com/update", params, new HttpCallback() {  
 @Override  
 public void onSuccess(String t) {  
 Loger.debug("请求到的数据:" + t);  
 }  
});

比起 **入门** 章节讲述的网络请求，你可能希望有更多的需求

构建网络请求

HttpParams params = new HttpParams();  
  
//同之前的设计，传递 http 请求头可以使用 putHeaders()  
params.putHeaders("cookie", "your cookie");  
params.putHeaders("User-Agent", "rxvolley");   
  
//传递 http 请求参数可以使用 put()  
params.put("name", "kymjs");  
params.put("age", "18");  
  
//http请求的回调，内置了很多方法，详细请查看源码  
//包括在异步响应的onSuccessInAsync():注不能做UI操作  
//网络请求成功时的回调onSuccess()  
//网络请求失败时的回调onFailure():例如无网络，服务器异常等  
HttpCallback callback = new HttpCallback(){  
 @Override  
 public void onSuccessInAsync(byte[] t) {  
 }  
 @Override  
 public void onSuccess(String t) {  
 }  
 @Override  
 public void onFailure(int errorNo, String strMsg) {  
 }  
}  
  
ProgressListener listener = new ProgressListener(){  
 /\*\*  
 \* @param transferredBytes 进度  
 \* @param totalSize 总量  
 \*/  
 @Override  
 public void onProgress(long transferredBytes, long totalSize){  
 }  
}  
  
new RxVolley.Builder()  
 .url("http://www.kymjs.com/rss.xml") //接口地址   
 //请求类型，如果不加，默认为 GET 可选项：   
 //POST/PUT/DELETE/HEAD/OPTIONS/TRACE/PATCH   
 .httpMethod(RxVolley.Method.GET)   
 //设置缓存时间: 默认是 get 请求 5 分钟, post 请求不缓存   
 .cacheTime(6)   
 //内容参数传递形式，如果不加，默认为 FORM 表单提交，可选项 JSON 内容  
 .contentType(RxVolley.ContentType.FORM)  
 .params(params) //上文创建的HttpParams请求参数集  
 //是否缓存，默认是 get 请求 5 缓存分钟, post 请求不缓存  
 .shouldCache(true)   
 .progressListener(listener) //上传进度  
 .callback(callback) //响应回调  
 .encoding("UTF-8") //编码格式，默认为utf-8  
 .doTask(); //执行请求操作

对 RxJava 的支持

```java  
public class Result {  
 public String url;  
 public byte[] data;  
 public VolleyError error;  
 public Map<String, String> headers;  
 public int errorCode;  
}

执行一次请求，并返回 Observable<Result>

Observable<Result> observable = new RxVolley.Builder()  
 .url("http://www.kymjs.com/rss.xml")  
 //default GET or POST/PUT/DELETE/HEAD/OPTIONS/TRACE/PATCH  
 .httpMethod(RxVolley.Method.POST)   
 .cacheTime(6) //default: get 5min, post 0min  
 .params(params)  
 .contentType(RxVolley.ContentType.JSON)  
 .getResult(); // 使用getResult()来返回RxJava数据类型  
  
//当拿到 observable 对象后，你可以设置你自己的 subscriber   
observable.subscribe(subscriber);

完整的使用示例

public class MainActivity extends AppCompatActivity {  
  
 private Subscription subscription;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_main);  
  
 Observable<Result> observable = new RxVolley.Builder()  
 .url("http://kymjs.com/feed.xml")  
 .contentType(RxVolley.ContentType.FORM)  
 .getResult();  
  
 subscription = observable  
 .map(new Func1<Result, String>() {  
 @Override  
 public String call(Result result) {  
 return new String(result.data);  
 }  
 })  
 .subscribeOn(Schedulers.io())  
 .observeOn(AndroidSchedulers.mainThread())  
 .subscribe(new Subscriber<String>() {  
 @Override  
 public void onCompleted() {  
 Log.i("kymjs", "======网络请求结束");  
 }  
  
 @Override  
 public void onError(Throwable e) {  
 Log.i("kymjs", "======网络请求失败" + e.getMessage());  
 }  
  
 @Override  
 public void onNext(String s) {  
 Log.i("kymjs", "======网络请求" + s);  
 }  
 });  
 }  
  
 @Override  
 protected void onDestroy() {  
 super.onDestroy();  
 if (subscription != null && subscription.isUnsubscribed()) {  
 subscription.unsubscribe();  
 }  
 }  
}

自定义请求

也许你是 Volley 的重度使用者(就像我)，那么你一定是因为 Volley 自由的扩展性而爱上它的。

你可以通过创建一个Request<?>的子类，自由配置请求策略，缓存策略，数据传输加密，重试策略等。

最后通过

RxVolley.Builder().setRequest(yourRequest).doTask();

去执行你的自定义 Request

一个典型自定义Request的示例：

/\*\*  
 \* Form表单形式的Http请求  
 \*/  
public class FormRequest extends Request<byte[]> {  
  
 private final HttpParams mParams;  
  
 public FormRequest(RequestConfig config, HttpParams params, HttpCallback callback) {  
 super(config, callback);  
 if (params == null) {  
 params = new HttpParams();  
 }  
 this.mParams = params;  
 }  
  
 @Override  
 public String getCacheKey() {  
 if (getMethod() == RxVolley.Method.POST) {  
 return getUrl() + mParams.getUrlParams();  
 } else {  
 return getUrl();  
 }  
 }  
  
 @Override  
 public String getBodyContentType() {  
 if (mParams.getContentType() != null) {  
 return mParams.getContentType();  
 } else {  
 return super.getBodyContentType();  
 }  
 }  
  
 @Override  
 public ArrayList<HttpParamsEntry> getHeaders() {  
 return mParams.getHeaders();  
 }  
  
 @Override  
 public byte[] getBody() {  
 ByteArrayOutputStream bos = new ByteArrayOutputStream();  
 try {  
 mParams.writeTo(bos);  
 } catch (IOException e) {  
 Loger.debug("FormRequest#getBody()--->IOException writing to ByteArrayOutputStream");  
 }  
 return bos.toByteArray();  
 }  
  
 @Override  
 public Response<byte[]> parseNetworkResponse(NetworkResponse response) {  
 return Response.success(response.data, response.headers,  
 HttpHeaderParser.parseCacheHeaders(getUseServerControl(), getCacheTime(),  
 response));  
 }  
  
 @Override  
 protected void deliverResponse(ArrayList<HttpParamsEntry> headers, final byte[] response) {  
 if (mCallback != null) {  
 HashMap<String, String> map = new HashMap<>(headers.size());  
 for (HttpParamsEntry entry : headers) {  
 map.put(entry.k, entry.v);  
 }  
 mCallback.onSuccess(map, response);  
 }  
 }  
  
 @Override  
 public Priority getPriority() {  
 return Priority.IMMEDIATE;  
 }  
}

文件(图片)下载

利用 RxVolley 的自定义请求，在库中内置了文件下载功能。你可以使用

//下载进度(可选参数，不需要可不传)  
listener = new ProgressListener() {  
 @Override  
 public void onProgress(long transferredBytes, long totalSize) {  
 Loger.debug(transferredBytes + "======" + totalSize);  
 }  
}  
  
//下载回调，内置了很多方法，详细请查看源码  
//包括在异步响应的onSuccessInAsync():注不能做UI操作  
//下载成功时的回调onSuccess()  
//下载失败时的回调onFailure():例如无网络，服务器异常等  
HttpCallback callback = new HttpCallback(){  
 @Override  
 public void onSuccessInAsync(byte[] t) {  
 }  
 @Override  
 public void onSuccess(String t) {  
 }  
 @Override  
 public void onFailure(int errorNo, String strMsg) {  
 }  
}  
  
RxVolley.download(FileUtils.getSDCardPath() + "/a.apk",  
 "https://www.oschina.net/uploads/osc-android-app-2.4.apk",  
 listener, callback);

download()原型

既然说了下载功能是利用 RxVolley 的自定义请求创建的，不妨看看他的方法实现：

/\*\*  
 \* 下载  
 \*  
 \* @param storeFilePath 本地存储绝对路径  
 \* @param url 要下载的文件的url  
 \* @param progressListener 下载进度回调  
 \* @param callback 回调  
 \*/  
 public static void download(String storeFilePath, String url, ProgressListener  
 progressListener, HttpCallback callback) {  
 RequestConfig config = new RequestConfig();  
 config.mUrl = url;  
 FileRequest request = new FileRequest(storeFilePath, config, callback);  
 request.setOnProgressListener(progressListener);  
 new Builder().setRequest(request).doTask();  
 }

更多可选设置

理论上来说，一切的请求设置都可以通过自定义 Request 来完成。

但是，如果你和我一样是个懒人，当然更希望这些早就有人已经做好了。

设置文件缓存的路径

默认的文件缓存路径是在SD卡根目录的 /RxVolley 文件夹下，你可以通过如下语句设置你的 cacheFolder

RxVolley.setRequestQueue(RequestQueue.newRequestQueue(cacheFolder));

需要注意的是，setRequestQueue 方法必须在 RxVolley.Build() 方法执行之前调用，也就是在使用 RxVolley 以前先设置配置信息。建议在 Application 类中完成这些设置。

Https设置

如果不设置，默认信任全部的https证书。可以传入自定义 SSLSocketFactory

RxVolley.setRequestQueue(RequestQueue.newRequestQueue(cacheFolder), new HttpConnectStack(null, sslSocketFactory));

需要注意的是，setRequestQueue 方法必须在 RxVolley.Build() 方法执行之前调用，也就是在使用 RxVolley 以前先设置配置信息。建议在 Application 类中完成这些设置。

一个自定义设置SSLSocketFactory的相关示例：

//下载的证书放到项目中的assets目录中  
InputStream ins = context.getAssets().open("app\_pay.cer");   
CertificateFactory cerFactory = CertificateFactory  
 .getInstance("X.509");  
Certificate cer = cerFactory.generateCertificate(ins);  
KeyStore keyStore = KeyStore.getInstance("PKCS12", "BC");  
keyStore.load(null, null);  
keyStore.setCertificateEntry("trust", cer);  
  
SSLSocketFactory socketFactory = new SSLSocketFactory(keyStore);  
  
RxVolley.setRequestQueue(RequestQueue.newRequestQueue(RxVolley.CACHE\_FOLDER), new HttpConnectStack(null, sslSocketFactory));

Build()中的可选设置

• 详细请参阅 RxVolley$Builder 类中代码。

//请求超时时间   
timeout()   
  
//为了更真实的模拟网络,如果读取缓存,延迟一段时间再返回缓存内容   
delayTime()   
  
//缓存有效时间,单位分钟   
cacheTime()   
  
//使用服务器控制的缓存有效期,即cookie有效期   
//(如果使用服务器端控制,则无视#cacheTime())   
useServerControl()   
  
//启用缓存   
shouldCache()   
  
//重连策略,Volley默认的重连策略是timeout=3000，重试1次   
retryPolicy()

常见问题

无对应key接收的参数

后台可以用数组接收，例如php代码实现：

<?php  
 foreach($\_POST as $key => $val){  
 if(is\_array($val)) {  
 foreach ($val as $v2) {  
 echo "数组key中的数据：$v2<br>";  
 }  
 } else {  
 echo "key为$key 的数据：$val<br>";  
 }  
 }  
?>

客户端传值时需要这么写：

//其中name[]，可以写为任意字符，但必须以[]结尾  
HttpParams params = new HttpParams();  
 params.put("name[]", "hello");  
 params.put("name[]", "hello2");  
 params.put("name[]", "hello3");  
 params.put("name2", "hello3333");  
 RxVolley.post("http://xxx.xxx.xxx/users.php", params, new HttpCallback() {  
 });

ProGuard 配置

For RxVolley

-dontwarn com.kymjs.rxvolley.

-keep class com.kymjs.rxvolley. {\*;}

RxVolley自定义扩展

Cookie持久化封装

RxVolley 默认对于 cookie 的操作是会从 HttpCallback 中返回 cookie，需要手动保存到本地。

如果你希望框架能够自动存储 cookie，可以这么做：

按需要选择继承FormRequest或者JsonRequest (直接继承 Request 类也可以，但是复杂) 并重写

@Override  
protected void deliverResponse(ArrayList<HttpParamsEntry> headers, final byte[] response) {  
 if (mCallback != null) {  
 HashMap<String, String> map = new HashMap<>(headers.size());  
 for (HttpParamsEntry entry : headers) {  
 map.put(entry.k, entry.v);  
 }  
 mCallback.onSuccess(map, response);  
 }   
}

逻辑如上述代码，其中的map即包含了服务器返回的cookie，可以做你自己的操作了。

最终执行你的自定义 Request

new RxVolley.Builder().setRequest(xxxxx).doTask();

在传递 Cookie 作为请求头的时候，建议写一个工具类，例如

public static HttpParams getHttpParams() {  
 HttpParams params = new HttpParams();  
 params.putHeader("cookie");  
 return params;  
}

使用 OkHttp

使用 OkHttp 替代 HttpUrlconnection

Volley 允许你创建自己的网络请求执行器，执行器需要实现IHttpStack接口

RxVolley 的 okhttp module 已经有了使用 OkHttp 作为请求执行器的实现。

你可以使用如下代码设置，依旧需要注意的是，setRequestQueue 方法必须在 RxVolley.Build() 方法执行之前调用，也就是在使用 RxVolley 以前先设置配置信息。建议在 Application 类中完成这些设置。

RxVolley.setRequestQueue(RequestQueue.newRequestQueue(RxVolley.CACHE\_FOLDER, new OkHttpStack(new OkHttpClient())));

使用 OkHttp 相关功能需要在你的 build.gradle 文件中加入

compile 'com.kymjs.rxvolley:okhttp:1.0.5'

历史版本

v1.0.X

Issue反馈