使用步骤如下：

1. 使用这几个框架需要配置环境：

//rxjava

compile 'io.reactivex.rxjava2:rxjava:2.0.1'

compile 'io.reactivex.rxjava2:rxandroid:2.0.1'

//retrofit

compile 'com.squareup.retrofit2:retrofit:2.1.0'

compile 'com.squareup.retrofit2:converter-gson:2.1.0'

compile 'com.jakewharton.retrofit:retrofit2-rxjava2-adapter:1.0.0'

compile 'com.squareup.okhttp3:okhttp:3.4.1'

compile 'com.squareup.okhttp3:logging-interceptor:3.4.1'

1. 定义一个接口，写请求

public interface IUserBiz

{

@GET("users")

Call<List<User>> getUsers();

//可以更多

}

1. 完成请求：创建一个retrofit，通过retrofit的create方法创建刚才写的请求接口，通过Call的enqueue（异步）或者execute（同步）进行网络请求

Retrofit retrofit = new Retrofit.Builder()

.baseUrl("http://192.168.31.242:8080/springmvc\_users/user/")

.addConverterFactory(GsonConverterFactory.create())

.build();

IUserBiz userBiz = retrofit.create(IUserBiz.class);

Call<List<User>> call = userBiz.getUsers();

call.enqueue(new Callback<List<User>>()

{

@Override

public void onResponse(Call<List<User>> call, Response<List<User>> response)

{

Log.e(TAG, "normalGet:" + response.body() + "");

}

@Override

public void onFailure(Call<List<User>> call, Throwable t)

{

}

});

常见参数的使用：

1. 请求的url需要动态附加路径的可用@path，例如

http://192.168.1.102:8080/springmvc\_users/user/zhy

//用于访问lmj的信息

http://192.168.1.102:8080/springmvc\_users/user/lmj

public interface IUserBiz

{

@GET("{username}")

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

}

注意：{username}通过{}当做占位符，而实际运行中会通过@PATH("username")所标注的参数进行替换。

1. 查询参数的设置可用@Query，例如

http://baseurl/users?sortby=username

http://baseurl/users?sortby=id

public interface IUserBiz

{

@GET("users")

Call<List<User>> getUsersBySort(@Query("sortby") String sort);

}

通过@Query(“字段名”)来表示

5. POST请求体的方式向服务器传入json字符串@Body

public interface IUserBiz

{

@POST("add")

Call<List<User>> addUser(@Body User user);

}

6. 表单的方式传递键值对@FormUrlEncoded

public interface IUserBiz

{

@POST("login")

@FormUrlEncoded

Call<User> login(@Field("username") String username, @Field("password") String password);

}

7.单文件上传@Multipart

public interface IUserBiz

{

@Multipart

@POST("register")

Call<User> registerUser(@Part MultipartBody.Part photo, @Part("username") RequestBody username, @Part("password") RequestBody password);

}

@SerializedName是gson提供的注解，将服务器的变量名注解出命名规范的变量名

例如：

@SerializedName("n")

private String userName;

参考：<http://www.cnblogs.com/newcaoguo/p/6104884.html>

Parcelable：序列化接口，永久性保存对象，可在网络间传递对象。Android特有，可用与intent传输和进程间通信。

使用方法：

1）implements Parcelable

2）重写writeToParcel方法，将你的对象序列化为一个Parcel对象，即：将类的数据写入外部提供的Parcel中，打包需要传递的数据到Parcel容器保存，以便从 Parcel容器获取数据

3）重写describeContents方法，内容接口描述，默认返回0就可以

4）实例化静态内部对象CREATOR实现接口Parcelable.Creator

5）需写一个将parcel导出的构造方法

public static final Parcelable.Creator<T> CREATOR

注：其中public static final一个都不能少，内部对象CREATOR的名称也不能改变，必须全部大写。需重写本接口中的两个方法：createFromParcel(Parcel in) 实现从Parcel容器中读取传递数据值，封装成Parcelable对象返回逻辑层，newArray(int size) 创建一个类型为T，长度为size的数组，仅一句话即可（return new T[size]），供外部类反序列化本类数组使用。