Retrofit的使用：

插件 GsonFormat的使用：

可一件把json数据生成类文件字段

使用步骤：File-setting-plungins-搜索GsonFormat，点击install，安装完需要重启下android studio

快捷键Alt+insert 弹窗选择GsonFormat或者右键generate-GsonFormat，粘json数据

1.入门使用：

添加依赖

compile 'com.squareup.retrofit2:retrofit:2.3.0'

compile 'com.squareup.retrofit2:converter-gson:2.3.0' //用于内部请求结果解析

定义一个接口，取名RetrofitService ：

public interface RetrofitService {

@GET("filter/search")

Call<Categories> getCategories(@Query("q") String name,

@Query("tag") String tag,

@Query("start") int start,

@Query("count") int count);

}

初始化：

public RetrofitService service ;

public RetrofitHttp() {

Retrofit retrofit = new Retrofit.Builder()

.baseUrl("http://app.fromfactory.club") // 请求的接口的baseurl

.addConverterFactory(GsonConverterFactory.create(new GsonBuilder().create()))

.build();

service = retrofit.create(RetrofitService.class);

}

public void getCategories() {

Call<Categories> call = service.getCategories();

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

@Override

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

Zlog.ii("lxm getCategories:" +response.body().toString());

}

@Override

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

}

});

}

最终访问的地址是http://app.fromfactory.clu/filter/search?q=abc &tag=&start=0&count=1

2.方法介绍

在方法的开头我们可以看到有个GET的注释，说明这个请求是GET方法，当然你也可以根据具体需要用POST、PUT、DELETE以及HEAD。他们的区别如下：

GET ----------查找资源（查）

POST --------修改资源（改）

PUT ----------上传文件（增）

DELETE ----删除文件（删）

HEAD--------只请求页面的首部

@Query(GET请求):

用于在url后拼接上参数，例如：

@GET("filter/search")

Call<Categories> getCategories(@Query("q") String name);//name由调用者传入

相当于：

@GET("filter/search?q=name")

Call<Categories> getCategories();

@QueryMap(GET请求):

当然如果入参比较多，就可以把它们都放在Map中，例如：

@GET("book/search")

Call<Categories> getCategories(@QueryMap Map<String, String> options);

@Path(GET请求):

用于替换url中某个字段，例如：

@GET("group/{id}/users")

Call<Categories> getCategories(@Path("id") int Id);

@Body(POST请求):

可以指定一个对象作为HTTP请求体,比如：

@POST("users/new")

Call<User> createUser(@Body User user);

它会把我们传入的User实体类转换为用于传输的HTTP请求体，进行网络请求。

@Field(POST请求):

用于传送表单数据：

@FormUrlEncoded

@POST("user/edit")

Call<User> updateUser(@Field("first\_name") String first, @Field("last\_name") String last);

注意开头必须多加上@FormUrlEncoded这句注释，不然会报错。表单自然是有多组键值对组成，这里的first\_name就是键，而具体传入的first就是值啦。

@ FieldMap (POST请求):

当传送的表单数据多时，也可以用map

@FormUrlEncoded

@POST("club\_factory/version/android")

Call<HttpJsonResult> updateVersion(@Query("version") String version\_id, @FieldMap Map<String,String> map);

@Header/@Headers(POST请求):

用于添加请求头部：

@GET("user")

Call<User> getUser(@Header("Authorization") String authorization)

表示将头部Authorization属性设置为你传入的authorization；当然你还可以用@Headers表示,作用是一样的比如：

@Headers("Cache-Control: max-age=640000")

@GET("user")

Call<User> getUser()

当然你可以多个设置：

@Headers({

"Accept: application/vnd.github.v3.full+json",

"User-Agent: Retrofit-Sample-App"

})

@GET("user")

Call<User> getUser()

3.retrofit的原理和与其他网路框架的比较