# 常用类与方法

## TextUtils

TextUtils.isEmpty()：判断是否为空字符串。

TextUtils.isDigitsOnly：判断是否只有数字。

TextUtils.equals()：判断字符串是否相等 。

TextUtils.getReverse()：获取字符串的倒序 。

TextUtils.getTrimmedLength():获取字符串的长度,忽略空格

TextUtils.htmlEncode():获取html格式的字符串

TextUtils.indexOf():获取字符串中第一次出现子字符串的字符位置

TextUtils.substring(int start ,int end)://截取字符串

TextUtils.split(String str,String temp):通过表达式截取字符串

## Gson

提供了fromJson() 和toJson() 两个直接用于解析和生成的方法，前者实现反序列化，后者实现了序列化。

同时每个方法都提供了重载方法;

1. 使用：fromJson(String jsonStr,Object .class);

将字符串转化为对应的bean，

eg：

1. Gson gson = new Gson();

int i = gson.fromJson("100", int.class); 转化为基本的数据类型

1. Gson gson = new Gson();

String jsonString = "{\"name\":\"怪盗kidou\",\"age\":24}";

User user = gson.fromJson(jsonString, User.class);转化bean类

1. toJson(Object object)将类转化为对应的json字符串
2. Gson gson = new Gson();

String jsonNumber = gson.toJson(100);

1. Gson gson = new Gson();

User user = new User("怪盗kidou",24);

String jsonObject = gson.toJson(user);

## HttpUtils

这个需要看使用的是什么框架，下面以retrofit2为例子

1. 设置基本参数

private static OkHttpClient okHttpClient = new OkHttpClient.Builder()

.connectTimeout(DEFAULT\_TIMEOUT, TimeUnit.SECONDS)

.writeTimeout(DEFAULT\_TIMEOUT, TimeUnit.SECONDS)

.connectionPool(new ConnectionPool(NetConfig.CONNECTION\_COUNT, NetConfig.CONNECTION\_TIME, TimeUnit.SECONDS))

// 设置8个同时连接的个数和每个保持时间为20s

.build();

1. private static Retrofit retrofit = new Retrofit.Builder()

.addConverterFactory(GsonConverterFactory.create())

.client(okHttpClient)

//设置服务器ip，必须依据“/”结尾

.baseUrl(NetConfig.URL\_REL)

.build();

1. private static RetrofitClientService retrofitClientService = retrofit.create(RetrofitClientService.class);
2. 对应的service

@FormUrlEncoded

@POST

Call<BaseResponse> executePost(@Url String url, @FieldMap HashMap<String, String> map);

1. 执行post请求

public static void executePost(String url, HashMap<String, String> map, final HttpListener listener) {

Call<BaseResponse> baseCall =retrofitClientService.

executePost(url,map);

baseCall.enqueue(callback);

1. private static Callback<BaseResponse> callback = new Callback<BaseResponse>() {

@Override

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

@Override

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

};

## Activity

onCreate()：创建

onDestroy:销毁

Fragment

碎片

onCreate();创建

onAcivity():活动创建时