# 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);

在实际项目中，经常会再次包装，整理成JsonUtils工具类。

## Map

（key，value）形式存储数据

|  |  |  |  |
| --- | --- | --- | --- |
|  | 优点 | 缺点 | 特性 |
| HashMap | 访问速度快 | 非同步 | key，value都可以为null，无序 |
| TreeMap | 有序，默认为升序 | 非同步 | key不能为null |
| HashTable | 线程同步 | 写入速度慢 | key，value都不能为null |
| LinkedHashMap | 有序，按照插入的顺序 | 读取稍慢 | key，value都可以为null |

方法：

1. put（key，value），插入
2. containsKey(key)，判断是否存在key
3. get（key），读写
4. remove（key），删除
5. clear()，清除map。
6. 转换为list

List<Map.Entry<String, String>> list = new ArrayList<Map.Entry<String, String>>(map.entrySet());

1. 遍历：

使用entrySet()遍历，最优

for (Map.Entry<String, String> entry : map.entrySet()) { System.out.println(entry.getKey() + " ：" + entry.getValue());}

## List

1. 添加方法是：.add(e)；
2. 获取方法是：.get(index)；
3. 删除方法是：.remove(index)；
4. 按照索引删除；.remove(Object o)；
5. list中是否包含某个元素；

方法：.contains（Object o）； 返回true或者false

1. list中根据索引将元素数值改变(替换)；

注意 .set(index, element); 和 .add(index, element); 的不同；

1. 利用list中索引位置重新生成一个新的list（截取集合）；

方法： .subList(fromIndex, toIndex)；该方法得到list中的元素数的和

1. 遍历：for（Object object：list）
2. 排序：

Collections.sort(list); 必须实现Comparable<T>接口，然后实现

compareTo（）方法，必须int，Long已经实现该方法

sort(List<T> list, Comparator<T> c)，第二个参数：传入类型，实现其compare方法