JSON, JavaScript object notation（JavaScript 对象表示法）一个用于数据交换的文本格式，是一种数据格式。与之类似的还有XML，YAML，但JSON的语法最简单。

////////////////////////////////////////////////////////////////////////////////////////////

JSON内部数据以名值对的形式存储

"name"："daring jiang",

"age"：22

JSON是一个树状结构，他只有六种数据格式（话说Java我好像也只用6种。。。）

* null： 就是null
* boolean： true/false
* number： 这个number会有点烦。由四部分组成。负号，整数部分，小数部分，科学计数法部分。这里面只有整数部分是必须的，其他可以忽略。下面挨个讲。

~负号，看清楚，负号，不是老子打错了。JSON不允许有正号。。。。。

~整数部分，不允许以0开头，除非就一个0.

~小数部分，没什么好讲的。

~科学计数法，e/E -/+ digit，该格式要是忘了就去百度。

* string： "....."
* array： [ .... ]
* object： { .... }

作为一个C语言家族学习者，嵌套总不用交了吧！

////////////////////////////////////////////////////////////////////////////////////////////

~Java JSON APIs

　　当JSON开始流行时，Java并没有一个标准的JSON生成器或分析器。因此，Java开发者们都只能依赖他人编写的开源Java JSON API。当然，现在Java已经有意识的去编写官方API，不过，还没完成。

　　现在用的比较多的是下面这几个。

* + - Jackson
    - GSON
    - Boon

　　当然了，要是有兴趣的话自己写一个也没问题啊，造轮子嘛，知乎上我关注的某个大V就正在写。但这边我选择GSON。

////////////////////////////////////////////////////////////////////////////////////////////

~GSON

　　这个API是谷歌创造的，这也是‘G’的由来。GSON非常的灵活，只不过GSON运行速度不是特别快

GSON包含三个解析器与生成器。

* GSON

// 将JSON转换为对应的Java对象

// 从Java对象处生成JSON，所以即是解析器又是生成器

* GSON JsonReader

// 序列化？

* GSON JsonParser

// 将JSON转换为对应的树形结构

下面挨个讲。

一．Gson

创建Gson对象有两种方式。

* Gson gson = new Gson();
* GsonBuilder builder= new GsonBuilder();

Gson gson = builder.create();

1. toJson()&fromJson()

String json = “.....”;

Gson gson = new Gson();

MyObject myObject = gson.fromJson(json, MyObject.class);

你看，重点就在于这个fromJson（）方法。第一个参数是JSON文件，这边是一个String类型的玩意，下一次可能是其他的什么。第二个参数是Class文件。所以，本质上，这个方法就是用Gson对象来创建目标类的对象，并将JSON中的数据导进去。因此，目标类最好没用构造函数，别问为什么，我也不晓得。

把普通类对象转换为JSON就简单多了，这里引用上面的数据。

String json = gson.toJson(myObject);

但碰到泛型情况就不一样了，List<MyObject>，它里面至少有两种类型，因为MyObject内部搞不好还有其他自定义的类。

**Type listType = new TypeToken<List<String>>() {}.getType();**

**List<String> target = new LinkedList<String>();**

**target.add("blah");**

**Gson gson = new Gson();**

**String json = gson.toJson(target, listType);**

**List<String> target2 = gson.fromJson(json, listType);**

这是官方文档给的例子，但我始终没理解那个空匿名内部类是怎么起作用的，果然，还差的远啊。

1. 美化输出结果

但是，用默认构造函数，也就是Gson()；创建的Json文件，它打印时会将数据紧密的排列到一起。很难读，也很丑，就像没有CSS的HTML，不是人看的。

{“name”：“时间的决定”，“price”：12.8} // 多丑

好在GsonBuilder类有一个方法setPrettyPrinting()。

new GsonBuilder().setPrettyPrinting().create();这种模式下创建的Gson对象会美化打印的Json对象，让人如沐春风。

1. Exclude elements

有时候不需要把所有的数据都转换为Json（那样人家岂不是一点隐私都没了\_( : 3 ? L)\_）

* transient 关键字，最简单省事。
* @Expose（serialize = false， deserialize = false） 加个注解

### GsonBuilder.setExclusionStrategies()；自行Google

1. 序列化null值

默认构造函数构建的Gson对象不能序列化空值，（是是是，以后都用GsonBuilder行了吧）。

GsonBuilder builder = new GsonBuilder();

builder.serializeNulls();

//我发现啊，那些个额外功能，都在GsonBuilder的成员方法中

1. Custom Serialization and Deserialization

// （唉，你硬要告诉我这是自定义序列化的意思我也没办法啊），这话和标题没关系，就是想说。不过，还真是没想到，custom serialization 是自定义序列化的意思。

简单来讲，就是将Java值装换为自定义的Json，又或者，将自定义的Json装换为Java值。不过，这个功能，上面不是讲解过了吗？toJson（）和fromJson（）。

Fuck，完全理解错了。确实是将Java值装换为Json数据，但是是按照自己定义的方式来，反序列化同理。挺强大的啊，可惜我现在用不到，不过可以加到loading里。

1. JsonReader

JsonReader类用来 流化 Json数据。这句话是我讲的，不清楚对不对，原文如下。JsonReader类使开发人员可以像读“流”一样去读取Json字符串与Json文件。如果说Gson类使创建器的话，JsonReader就是工具吧。

1. 创建一个JsonReader

这个简单，比Gson类简单。

JsonReader jsonReader = new JsonReader（new StringReader（s））；

就是一个带参构造函数，这个参数必须是Reader类型，这里的例子使用了StringReader。（因为Json串一般都是字符串类型吗？）

2. 迭代

这个迭代看的我很不舒服，多查查官方文档吧。

当你创建了一个JsonReader对象时，你就可以通过你传给JsonReader构造函数的Reader对象来迭代Json tokens（象征，令牌？），这话，好高深啊。今天晚上我需要好好看看枚举类型

1. JsonParser

JsonParser类会分析Json字符串或流，并转换为一个由Java对象组成的树型结构。那么问题来了，什么是Java对象树形结构？

1. 创建JsonParser

JsonParser jsonParser = new JsonParser（）；

// 搞定！

1. 分析json并转换为树形结构

JsonParser parser = new JsonParser（）；

JsonElement jsonTree = parser.parse(json)； // 所以JsonElement类又是什么啊！

1. 迭代树形结构

这边的迭代基本是针对JsonElement类的成员方法，所以要多查官方文档，还有就是，我隐隐的有种预感，我暂时用不到。

* json转map，map转json