Gson 是Google公司发布的一个开放源代码的Java库，主要作用是序列化或反序列化java对象。

Json数据的四种类型：

Jsonprimitive 例如一个字符串或整型 “sss” 或1.22

Jsonobject 一个键值对的集合{"name":"sun","sex":"Female","married":false}

Jsonarray jsonobject的集合，元素可以是四种类型中的某一种或几种。

JsonNull null

## Gson解决的问题：

快速实现java对象和json之间的相互转换。

处理null的情况：

序列化时，如果对象的某个字段为null，不会输出到寄送字符串中。

反序列化的时候，某个字段在json字符串中找不到对应的值，就会被赋值为null

Javabean （java中标准）

具有无参构造、属性私有、提供属性的public get set方法。

一般我们会添加有参构造，这是必须显式提供无参构造，添加tostring() 便于调试。

## Gson创建对象的方式：

New Gson（） 输出格式默认，比如serializeNulls会是false

即序列化时值为null的时候不输出该字段。

new GsonBuilder().setDateFormat(sdf).serializeNulls().create()；可设置序列化反序列化时 对日期类型的数据输出格式和null的处理方式。Serializenulls（）调用后表示遇到遇到null时，正常进行打印。

# 序列化；

Jsonserializer是一个接口，当我们需要定制自己的序列化要求是，实现这个类重写serialize方法。

public class BookSerialiser implements JsonSerializer {

@Override

public JsonElement serialize(final Book book, final Type typeOfSrc, final JsonSerializationContext context) {

final JsonObject jsonObject = new JsonObject();

jsonObject.addProperty("title", book.getTitle());

jsonObject.addProperty("isbn-10", book.getIsbn10());

jsonObject.addProperty("isbn-13", book.getIsbn13());

final JsonArray jsonAuthorsArray = new JsonArray();

for (final String author : book.getAuthors()) {

final JsonPrimitive jsonAuthor = new JsonPrimitive(author);

jsonAuthorsArray.add(jsonAuthor);

}

jsonObject.add("authors", jsonAuthorsArray);

return jsonObject;

}

}

配置：

final GsonBuilder gsonBuilder = new GsonBuilder();

gsonBuilder.registerTypeAdapter(Book.class, new BookSerialiser());

gsonBuilder.setPrettyPrinting();

final Gson gson = gsonBuilder.create();

## 反序列化定制自己的和序列化基本一致，只是实现接口不同，jsonDeserializer。

## TypeAdapter

采用流的方式进行序列化和反序列化，效率更高。

Typetoke的使用：

在解析数组时，如果整个json只包含一个数组且没有其他字段，可以使用typetoke

用法：

Typetoke tt = new typetoke<list<object>>();

New gson().fromgson(targetGson,tt.gettype)

当Json数据不只包含数组还存在其他参数时，想要解析数组下的数据就需要定位到那个数组的节点上，可以通过JsonObject先去定位到数组的节点，再去得到数组下的数据。

