# 解析Json的方法

java对象转json格式就是两种方式，一种是框架，一种是用一些开源的封装好的json jar包。

springmvc是利用@ResponseBody这个注解将数据转化为json。

# Fastjson

## 基本概念

Fastjson API入口类是com.alibaba.fastjson.JSON，常用的序列化操作都可以在JSON类的静态方法直接完成，JSONObject类继承自JSON类。

## maven依赖

<dependency>

<groupId>com.alibaba</groupId>

<artifactId>fastjson</artifactId>

<version>1.2.3</version>

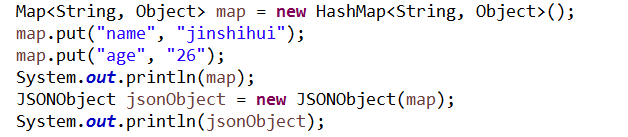
</dependency>

## 构造函数

### JSONObject(Map<String, Object> map);

//将map集合转换为JSONObject对象

示例：



输出：

{age=26, name=jinshihui}

{"age":"26","name":"jinshihui"}

## 方法

### public static final Object parse(String text);

//把JSON文本字符串解析为JSONObject或者JSONArray对象

说明：解析后json字符串中的顺序发生变化

### public static final JSONObject parseObject(String text);

//把JSON文本字符串解析成JSONObject对象（说明：json字符串中的顺序发生变化） 这个常用

### public static final <T> T parseObject(String text, Class<T> clazz);

//把JSON文本字符串parse为JavaBean

### public static final JSONArray parseArray(String text);

// 把JSON字符串parse成JSONArray

### public static final <T> List<T> parseArray(String text, Class<T> clazz);

//把JSON字符串parse成List集合

### public static final String toJSONString(Object object);

//将对象序列化为JSON文本字符串，这个常用。

### public static final String toJSONString(Object object, boolean prettyFormat);

//将JavaBean序列化为带格式的JSON文本

### public static final Object toJSON(Object javaObject);

//将Java对象转换为JSONObject或者JSONArray对象

### public Object get(String name)

//获取指定名称的数据值

说明：

SerializeWriter：相当于StringBuffer

JSONArray：相当于List<Object>

JSONObject：相当于Map<String, Object>

JSON反序列化没有真正数组，本质类型都是List<Object>

## 示例

### JavaBean转Json

@Test

public void BeanToJson(){

Person p1=new Person();

p1.setId(0);

p1.setName("nike");

String str = JSON.toJSONString(p1);

System.out.println(str);

}

### List转Json

@Test

public void listToJson(){

List <Person>list=new ArrayList<>();

Person p1=new Person();

p1.setId(0);

p1.setName("nike");

Person p2=new Person();

p2.setId(1);

p2.setName("linda");

list.add(p2);

list.add(p1);

String str = JSON.toJSONString(list);

System.out.println(str);

}

### Map转Json

@Test

public void mapToJson(){

Map<String, Object> map=new HashMap<String, Object>();

List <Person>list=new ArrayList<>();

Person p1=new Person();

p1.setId(0);

p1.setName("nike");

Person p2=new Person();

p2.setId(1);

p2.setName("linda");

list.add(p2);

list.add(p1);

map.put("list", list);

map.put("total", 1);

String str = JSON.toJSONString(map);

System.out.println(str);

}

# Gson

Gson 是google解析Json的一个开源框架,同类的框架fastJson,Jackson等。

## maven依赖

<dependency>

<groupId>com.google.code.gson</groupId>

<artifactId>gson</artifactId>

<version>2.8.5</version>

</dependency>

## Gson类

### 方法

#### String toJson(Object src)

//将java对象转换为json字符串

#### Object fromJson(String,Class)

//将json字符串转换为javabean对象

## JavaBean转换为json字符串

使用了toJson()方法



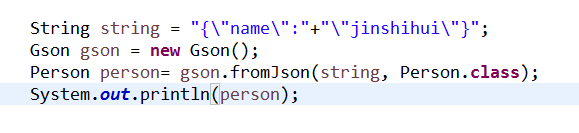
## List Map转换为json串

使用了toJson()方法



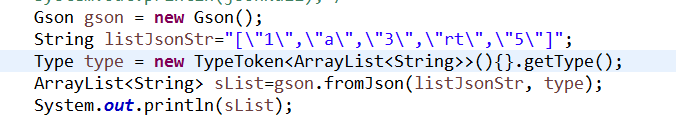
## Json字符串转换JavaBean

使用了fromJson()方法



## Json字符串转换List

使用了fromJson()方法



## JsonPrimitive

  JsonPrimitive非常有意思,我们知道如果json转换成字符串可能包含引号的转义，但是通过JsonPrimative我们可以获得转义的字符串。

### 示例代码



## JsonObject类

### 基本概念

表示Json中对象类型的类。对象由名称 - 值对组成，其中名称是字符串，值是{@link JsonElement}的任何其他类型。这允许创建JsonElements树。 此对象的成员元素按照添加顺序进行维护。

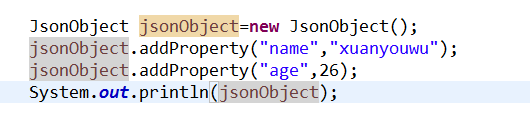
### 方法

#### public void add(String property, JsonElement value)

#### public void addProperty(String property, String value)

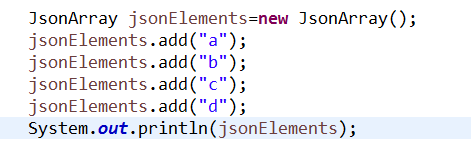
#### public JsonElement get(String memberName)

### 示例代码



输出：{"name":"xuanyouwu","age":26}

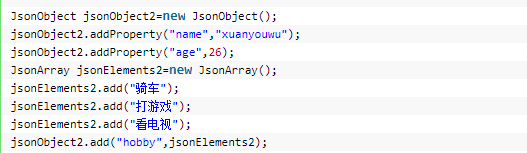
## 创建JsonArray



输出：["a","b","c","d"]

## JsonObject 嵌套数组或者说嵌套JsonArray

通过JsonObject的add(key,JsonElement)可以为jsonObject 添加一个数组的字段



输出：

{"name":"jinshihui","age":26,"hobby":["骑车","打游戏","看电视"]}

## Gson注解

### ****@SerializedName****

POJO的生成与解析可以看出json的字段和值的名称和类型是一一对应的，但也有一定容错机制，但有时候也会出现一些不和谐的情况，如：

期望的json格式：{"name":"张三","age":24,"emailAddress":"zhangsan@ceshi.com"}

实际：{"name":"张三","age":24,"email\_address":"zhangsan@ceshi.com"}

Gson在序列化和反序列化时需要使用反射，一般各类库都将注解放到annotations包下，打开源码在com.google.gson包下有一个annotations，里面有一个SerializedName的注解类。对于json中email\_address这个属性对应POJO的属性则变成：

@SerializedName("email\_address")

public String emailAddress;

为POJO字段提供备选属性名：SerializedName注解提供了两个属性，上面用到了其中一个，别外还有一个属性alternate，接收一个String数组

注：alternate需要2.4版本

代码如下所示：

@SerializedName(value = "emailAddress", alternate = {"email", "email\_address"})

public String emailAddress;

//当三个属性(email\_address、email、emailAddress)都中出现任意一个时均可以得到正确的结果

//当多种情况同时出时，以最后一个出现的值为准。

Gson gson = new Gson();

String json = "{\"name\":\"张三kidou\",\"age\":24,\"emailAddress\":\"zhangsan@ceshi.com\",\"email\":\"zhangsan\_2@ceshi.com\",\"email\_address\":\"zhangsan\_3@ceshi.com\"}";

User user = gson.fromJson(json, User.class);

System.out.println(user.emailAddress); // zhangsan\_3@example.com

# JackSon解析JSON对象

## 需要的jar包

Jackson有1.x系列和2.x系列，2.x系列有3个jar包需要下载：  
jackson-core-2.2.3.jar（核心jar包）  
jackson-annotations-2.2.3.jar（该包提供Json注解支持）  
jackson-databind-2.2.3.jar

## 核心类

ObjectMapper是JSON操作的核心，Jackson的所有JSON操作都是在ObjectMapper类中实现，ObjectMapper有多个JSON序列化的方法，可以把JSON字符串保存在File、OutputStream等不同的介质中。

## ObjectMapper类

### 方法

#### void writeValue(File resultFile, Object value)

把value转成json序列，并保存到file文件中

#### void writeValue(OutputStream arg0, Object arg1)

把arg1转成json序列，并保存到输出流中

#### byte[] writeValueAsBytes(Object arg0)

把arg0转成json序列，并把结果输出成字节数组

#### String writeValueAsString(Object arg0)（重点）

把对象arg0转成json序列，并把结果输出成字符串

#### <T> T readValue(String content, JavaType valueType)（重点）

从给定的JSON内容字符串反序列化JSON内容(也就是将JSON字符串转换为Java类)

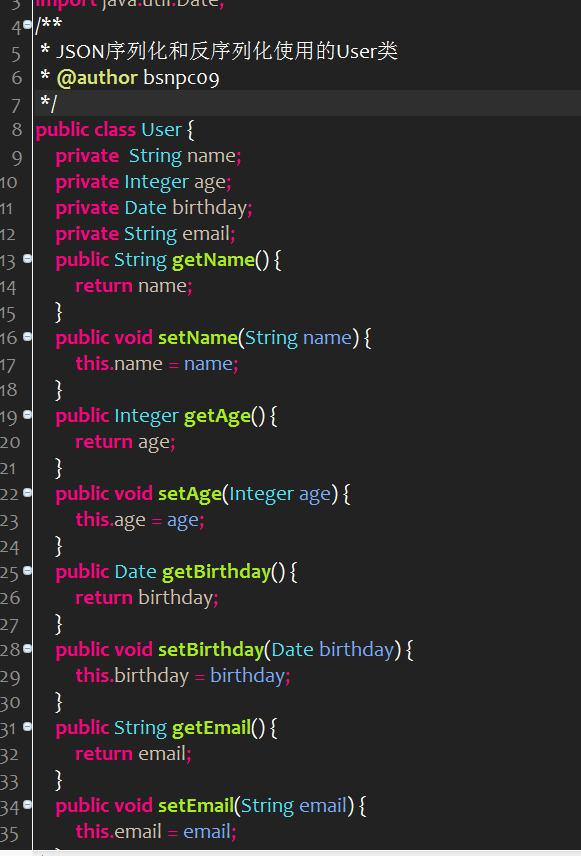
#### <T> T readValue(String content, TypeReference valueTypeRef)（重点）

从给定的JSON内容字符串反序列化JSON内容（也就是将JSON字符串转换为集合）

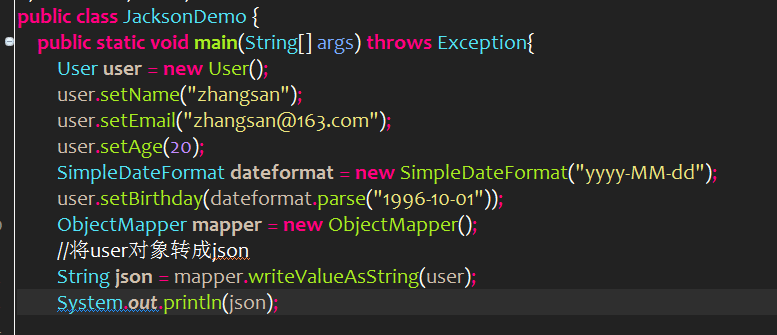
ObjectMapper支持从byte[]、File、InputStream、字符串等数据的JSON反序列化

## 示例代码

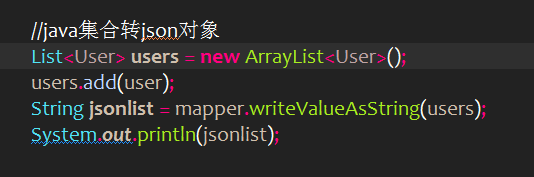
### （1）创建一个java实体类



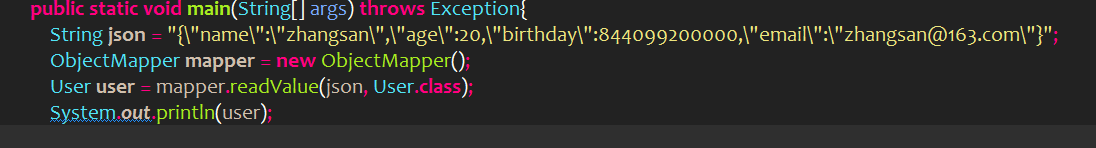
### （2）Java对象转json(json序列化)



### （3）Java集合转换为JSON

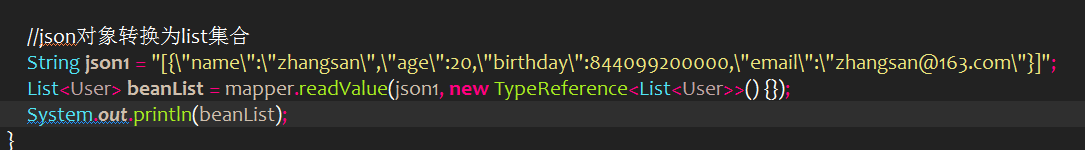


### （4）JSON转Java类（JSON反序列化）



说明：ObjectMapper支持从byte[]、File、InputStream、字符串等数据的JSON反序列化。

### （5）JSON转换为List集合



## JSON注解

Jackson提供了一系列注解，方便对JSON序列化和反序列化进行控制

### **常用的注解**

#### @JsonIgnore

此注解用于属性上，作用是进行JSON操作时忽略该属性。

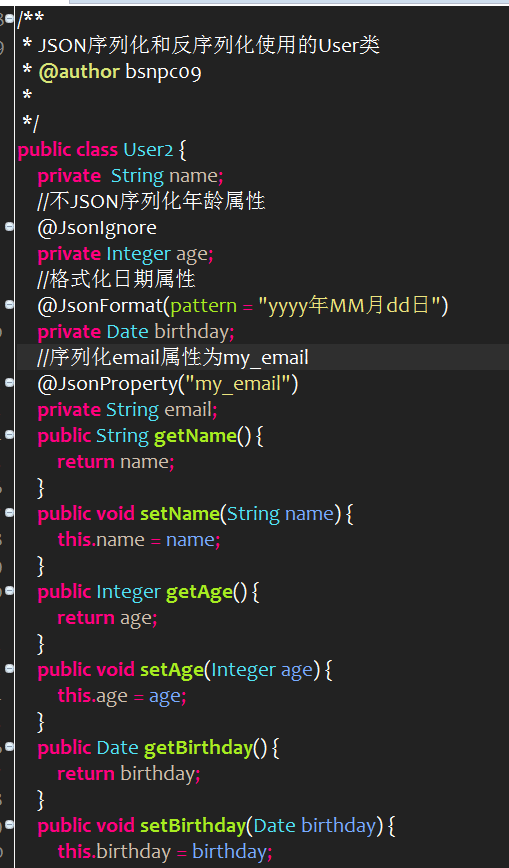
#### @JsonFormat

此注解用于属性上，作用是把Date类型直接转化为想要的格式，如@JsonFormat(pattern = "yyyy-MM-dd HH-mm-ss")

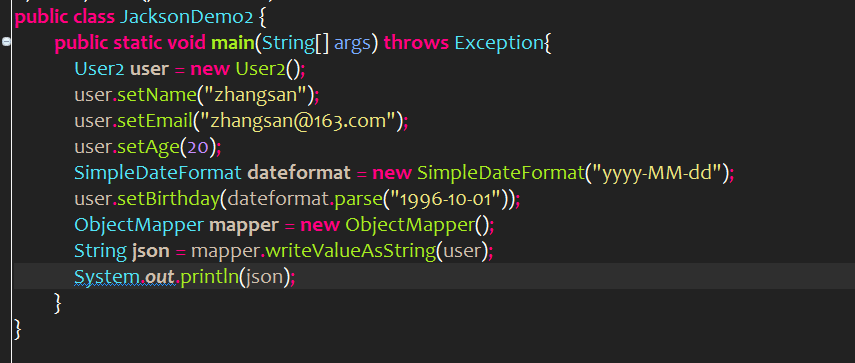
#### @JsonProperty

此注解用于属性上，作用是把该属性的名称序列化为另外一个名称，如把trueName属性序列化为name，@JsonProperty("name")

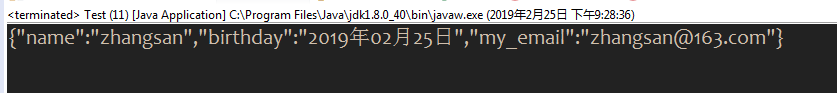
### 示例代码



Java类转JSON对象



输出的结果：



# 各个JSON框架的比较

## （1）json-lib（不用这个）

json-lib最开始的也是应用最广泛的json解析工具，json-lib 不好的地方确实是依赖于很多第三方包，包括commons-beanutils.jar，commons-collections-3.2.jar，commons-lang-2.6.jar，commons-logging-1.1.1.jar，ezmorph-1.0.6.jar，对于复杂类型的转换，json-lib对于json转换成bean还有缺陷，比如一个类里面会出现另一个类的list或者map集合，json-lib从json到bean的转换就会出现问题。json-lib在功能和性能上面都不能满足现在互联网化的需求。

## （2）开源的Jackson

相比json-lib框架，Jackson所依赖的jar包较少，简单易用并且性能也要相对高些。而且Jackson社区相对比较活跃，更新速度也比较快。Jackson对于复杂类型的json转换bean会出现问题，一些集合Map，List的转换出现问题。Jackson对于复杂类型的bean转换Json，转换的json格式不是标准的Json格式。

## （3）Google的Gson

Gson是目前功能最全的Json解析神器，Gson当初是应Google公司内部需求而由Google自行研发而来，但自从在2008年五月公开发布第一版后已被许多公司或用户应用。Gson的应用主要为toJson与fromJson两个转换函数，无依赖，不需要例外额外的jar，能够直接跑在JDK上。而在使用这种对象转换之前需先创建好对象的类型以及其成员才能成功的将JSON字符串成功转换成相对应的对象。类里面只要有get和set方法，Gson完全可以将复杂类型的json到bean或bean到json的转换，是JSON解析的神器。Gson在功能上面无可挑剔，但是性能上面比FastJson有所差距。

## （4）FastJson

Fastjson是一个Java语言编写的高性能的JSON处理器,由阿里巴巴公司开发。无依赖，不需要例外额外的jar，能够直接跑在JDK上。FastJson在复杂类型的Bean转换Json上会出现一些问题，可能会出现引用的类型，导致Json转换出错，需要制定引用。FastJson采用独创的算法，将parse的速度提升到极致，超过所有json库。

综上4种Json技术的比较，在项目选型的时候可以使用Google的Gson和阿里巴巴的FastJson两种并行使用，如果只是功能要求，没有性能要求，可以使用google的Gson，如果有性能上面的要求可以使用Gson将bean转换json确保数据的正确，使用FastJson将Json转换Bean。