# 简单介绍

选择一个合适的JSON库要从多个方面进行考虑：

* 字符串解析成JSON性能
* 字符串解析成JavaBean性能
* JavaBean构造JSON性能
* 集合构造JSON性能
* 易用性

# 常用库

## Gson

项目地址：https://github.com/google/gson

Gson是目前功能最全的Json解析神器，Gson当初是为因应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。 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完全可以实现复杂类型的json到bean或bean到json的转换，是JSON解析的神器。

在Java中的简单使用方法：

//1.创建对象

Gson gson = new Gson();

//2.向GSON中存数据

String str = gson.toJson(list);

//3.响应结果

response.getWriter().print(str);

//这是str的值：

"[{"id":5,"name":"纽约"},{"id":6,"name":"华盛顿"},{"id":7,"name":"洛杉矶"}]"

## FastJson

项目地址：https://github.com/alibaba/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。

### FastJson的介绍

JSON协议使用方便，越来越流行,JSON的处理器有很多,这里我介绍一下FastJson,FastJson是阿里的开源框架,被不少企业使用,是一个极其优秀的Json框架,Github地址: FastJson

### FastJson的特点:

* FastJson数度快,无论序列化和反序列化,都是当之无愧的fast
* 功能强大(支持普通JDK类包括任意Java Bean Class、Collection、Map、Date或enum)
* 零依赖(没有依赖其它任何类库)

### FastJson的简单说明:

FastJson对于json格式字符串的解析主要用到了下面三个类：

* JSON：fastJson的解析器，用于JSON格式字符串与JSON对象及javaBean之间的转换
* JSONObject：fastJson提供的json对象
* JSONArray：fastJson提供json数组对象

还在迷茫和彷徨吗,快上车,老司机带你飞!

## FastJson的用法

首先定义三个json格式的字符串

//json字符串-简单对象型

private static final String JSON\_OBJ\_STR = "{\"studentName\":\"lily\",\"studentAge\":12}";

//json字符串-数组类型

private static final String JSON\_ARRAY\_STR = "[{\"studentName\":\"lily\",\"studentAge\":12},{\"studentName\":\"lucy\",\"studentAge\":15}]";

//复杂格式json字符串

private static final String COMPLEX\_JSON\_STR = "{\"teacherName\":\"crystall\",\"teacherAge\":27,\"course\":{\"courseName\":\"english\",\"code\":1270},\"students\":[{\"studentName\":\"lily\",\"studentAge\":12},{\"studentName\":\"lucy\",\"studentAge\":15}]}";

### JSON格式字符串与JSON对象之间的转换

#### json字符串-简单对象型与JSONObject之间的转换

/\*\*

\* json字符串-简单对象型到JSONObject的转换

\*/

@Test

public void testJSONStrToJSONObject() {

JSONObject jsonObject = JSONObject.parseObject(JSON\_OBJ\_STR);

System.out.println("studentName: " + jsonObject.getString("studentName") + ":" + " studentAge: "

+ jsonObject.getInteger("studentAge"));

}

/\*\*

\* JSONObject到json字符串-简单对象型的转换

\*/

@Test

public void testJSONObjectToJSONStr() {

//已知JSONObject,目标要转换为json字符串

JSONObject jsonObject = JSONObject.parseObject(JSON\_OBJ\_STR);

// 第一种方式

String jsonString = JSONObject.toJSONString(jsonObject);

// 第二种方式

//String jsonString = jsonObject.toJSONString();

System.out.println(jsonString);

}

#### json字符串(数组类型)与JSONArray之间的转换

/\*\*

\* json字符串-数组类型到JSONArray的转换

\*/

@Test

public void testJSONStrToJSONArray() {

JSONArray jsonArray = JSONArray.parseArray(JSON\_ARRAY\_STR);

//遍历方式1

int size = jsonArray.size();

for (int i = 0; i < size; i++) {

JSONObject jsonObject = jsonArray.getJSONObject(i);

System.out.println("studentName: " + jsonObject.getString("studentName") + ":" + " studentAge: "

+ jsonObject.getInteger("studentAge"));

}

//遍历方式2

for (Object obj : jsonArray) {

JSONObject jsonObject = (JSONObject) obj;

System.out.println("studentName: " + jsonObject.getString("studentName") + ":" + " studentAge: "

+ jsonObject.getInteger("studentAge"));

}

}

/\*\*

\* JSONArray到json字符串-数组类型的转换

\*/

@Test

public void testJSONArrayToJSONStr() {

//已知JSONArray,目标要转换为json字符串

JSONArray jsonArray = JSONArray.parseArray(JSON\_ARRAY\_STR);

//第一种方式

String jsonString = JSONArray.toJSONString(jsonArray);

// 第二种方式

//String jsonString = jsonArray.toJSONString(jsonArray);

System.out.println(jsonString);

}

#### 复杂json格式字符串与JSONObject之间的转换

/\*\*

\* 复杂json格式字符串到JSONObject的转换

\*/

@Test

public void testComplexJSONStrToJSONObject() {

JSONObject jsonObject = JSONObject.parseObject(COMPLEX\_JSON\_STR);

String teacherName = jsonObject.getString("teacherName");

Integer teacherAge = jsonObject.getInteger("teacherAge");

System.out.println("teacherName: " + teacherName + " teacherAge: " + teacherAge);

JSONObject jsonObjectcourse = jsonObject.getJSONObject("course");

//获取JSONObject中的数据

String courseName = jsonObjectcourse.getString("courseName");

Integer code = jsonObjectcourse.getInteger("code");

System.out.println("courseName: " + courseName + " code: " + code);

JSONArray jsonArraystudents = jsonObject.getJSONArray("students");

//遍历JSONArray

for (Object object : jsonArraystudents) {

JSONObject jsonObjectone = (JSONObject) object;

String studentName = jsonObjectone.getString("studentName");

Integer studentAge = jsonObjectone.getInteger("studentAge");

System.out.println("studentName: " + studentName + " studentAge: " + studentAge);

}

}

/\*\*

\* 复杂JSONObject到json格式字符串的转换

\*/

@Test

public void testJSONObjectToComplexJSONStr() {

//复杂JSONObject,目标要转换为json字符串

JSONObject jsonObject = JSONObject.parseObject(COMPLEX\_JSON\_STR);

//第一种方式

//String jsonString = JSONObject.toJSONString(jsonObject);

//第二种方式

String jsonString = jsonObject.toJSONString();

System.out.println(jsonString);

}

### JSON格式字符串与javaBean之间的转换

#### json字符串-简单对象型与javaBean之间的转换

/\*\*

\* json字符串-简单对象到JavaBean之间的转换

\*/

@Test

public void testJSONStrToJavaBeanObj() {

//第一种方式

JSONObject jsonObject = JSONObject.parseObject(JSON\_OBJ\_STR);

String studentName = jsonObject.getString("studentName");

Integer studentAge = jsonObject.getInteger("studentAge");

//Student student = new Student(studentName, studentAge);

//第二种方式,使用TypeReference<T>类,由于其构造方法使用protected进行修饰,故创建其子类

//Student student = JSONObject.parseObject(JSON\_OBJ\_STR, new TypeReference<Student>() {});

//第三种方式,使用Gson的思想

Student student = JSONObject.parseObject(JSON\_OBJ\_STR, Student.class);

System.out.println(student);

}

/\*\*

\* JavaBean到json字符串-简单对象的转换

\*/

@Test

public void testJavaBeanObjToJSONStr() {

Student student = new Student("lily", 12);

String jsonString = JSONObject.toJSONString(student);

System.out.println(jsonString);

}

#### json字符串-数组类型与javaBean之间的转换

/\*\*

\* json字符串-数组类型到JavaBean\_List的转换

\*/

@Test

public void testJSONStrToJavaBeanList() {

//第一种方式

JSONArray jsonArray = JSONArray.parseArray(JSON\_ARRAY\_STR);

//遍历JSONArray

List<Student> students = new ArrayList<Student>();

Student student = null;

for (Object object : jsonArray) {

JSONObject jsonObjectone = (JSONObject) object;

String studentName = jsonObjectone.getString("studentName");

Integer studentAge = jsonObjectone.getInteger("studentAge");

student = new Student(studentName,studentAge);

students.add(student);

}

System.out.println("students: " + students);

//第二种方式,使用TypeReference<T>类,由于其构造方法使用protected进行修饰,故创建其子类

List<Student> studentList = JSONArray.parseObject(JSON\_ARRAY\_STR, new TypeReference<ArrayList<Student>>() {});

System.out.println("studentList: " + studentList);

//第三种方式,使用Gson的思想

List<Student> studentList1 = JSONArray.parseArray(JSON\_ARRAY\_STR, Student.class);

System.out.println("studentList1: " + studentList1);

}

/\*\*

\* JavaBean\_List到json字符串-数组类型的转换

\*/

@Test

public void testJavaBeanListToJSONStr() {

Student student = new Student("lily", 12);

Student studenttwo = new Student("lucy", 15);

List<Student> students = new ArrayList<Student>();

students.add(student);

students.add(studenttwo);

String jsonString = JSONArray.toJSONString(students);

System.out.println(jsonString);

}

#### 复杂json格式字符串与与javaBean之间的转换

/\*\*

\* 复杂json格式字符串到JavaBean\_obj的转换

\*/

@Test

public void testComplexJSONStrToJavaBean(){

//第一种方式,使用TypeReference<T>类,由于其构造方法使用protected进行修饰,故创建其子类

Teacher teacher = JSONObject.parseObject(COMPLEX\_JSON\_STR, new TypeReference<Teacher>() {});

System.out.println(teacher);

//第二种方式,使用Gson思想

Teacher teacher1 = JSONObject.parseObject(COMPLEX\_JSON\_STR, Teacher.class);

System.out.println(teacher1);

}

/\*\*

\* 复杂JavaBean\_obj到json格式字符串的转换

\*/

@Test

public void testJavaBeanToComplexJSONStr(){

//已知复杂JavaBean\_obj

Teacher teacher = JSONObject.parseObject(COMPLEX\_JSON\_STR, new TypeReference<Teacher>() {});

String jsonString = JSONObject.toJSONString(teacher);

System.out.println(jsonString);

}

### javaBean与json对象间的之间的转换

#### 简单javaBean与json对象之间的转换

/\*\*

\* 简单JavaBean\_obj到json对象的转换

\*/

@Test

public void testJavaBeanToJSONObject(){

//已知简单JavaBean\_obj

Student student = new Student("lily", 12);

//方式一

String jsonString = JSONObject.toJSONString(student);

JSONObject jsonObject = JSONObject.parseObject(jsonString);

System.out.println(jsonObject);

//方式二

JSONObject jsonObject1 = (JSONObject) JSONObject.toJSON(student);

System.out.println(jsonObject1);

}

/\*\*

\* 简单json对象到JavaBean\_obj的转换

\*/

@Test

public void testJSONObjectToJavaBean(){

//已知简单json对象

JSONObject jsonObject = JSONObject.parseObject(JSON\_OBJ\_STR);

//第一种方式,使用TypeReference<T>类,由于其构造方法使用protected进行修饰,故创建其子类

Student student = JSONObject.parseObject(jsonObject.toJSONString(), new TypeReference<Student>() {});

System.out.println(student);

//第二种方式,使用Gson的思想

Student student1 = JSONObject.parseObject(jsonObject.toJSONString(), Student.class);

System.out.println(student1);

}

#### JavaList与JsonArray之间的转换

/\*\*

\* JavaList到JsonArray的转换

\*/

@Test

public void testJavaListToJsonArray() {

//已知JavaList

Student student = new Student("lily", 12);

Student studenttwo = new Student("lucy", 15);

List<Student> students = new ArrayList<Student>();

students.add(student);

students.add(studenttwo);

//方式一

String jsonString = JSONArray.toJSONString(students);

JSONArray jsonArray = JSONArray.parseArray(jsonString);

System.out.println(jsonArray);

//方式二

JSONArray jsonArray1 = (JSONArray) JSONArray.toJSON(students);

System.out.println(jsonArray1);

}

/\*\*

\* JsonArray到JavaList的转换

\*/

@Test

public void testJsonArrayToJavaList() {

//已知JsonArray

JSONArray jsonArray = JSONArray.parseArray(JSON\_ARRAY\_STR);

//第一种方式,使用TypeReference<T>类,由于其构造方法使用protected进行修饰,故创建其子类

ArrayList<Student> students = JSONArray.parseObject(jsonArray.toJSONString(),

new TypeReference<ArrayList<Student>>() {});

System.out.println(students);

//第二种方式,使用Gson的思想

List<Student> students1 = JSONArray.parseArray(jsonArray.toJSONString(), Student.class);

System.out.println(students1);

}

#### 复杂JavaBean\_obj与json对象之间的转换

/\*\*

\* 复杂JavaBean\_obj到json对象的转换

\*/

@Test

public void testComplexJavaBeanToJSONObject() {

//已知复杂JavaBean\_obj

Student student = new Student("lily", 12);

Student studenttwo = new Student("lucy", 15);

List<Student> students = new ArrayList<Student>();

students.add(student);

students.add(studenttwo);

Course course = new Course("english", 1270);

Teacher teacher = new Teacher("crystall", 27, course, students);

//方式一

String jsonString = JSONObject.toJSONString(teacher);

JSONObject jsonObject = JSONObject.parseObject(jsonString);

System.out.println(jsonObject);

//方式二

JSONObject jsonObject1 = (JSONObject) JSONObject.toJSON(teacher);

}

## Jackson

项目地址：https://github.com/FasterXML/jackson

Jackson是当前用的比较广泛的，用来序列化和反序列化json的Java开源框架。Jackson社区相对比较活跃，更新速度也比较快， 从Github中的统计来看，Jackson是最流行的json解析器之一，Spring MVC的默认json解析器便是Jackson。

Jackson优点很多：

Jackson 所依赖的jar包较少，简单易用。

与其他 Java 的 json 的框架 Gson 等相比，Jackson 解析大的 json 文件速度比较快。

Jackson 运行时占用内存比较低，性能比较好

Jackson 有灵活的 API，可以很容易进行扩展和定制。

目前最新版本是2.9.4，Jackson 的核心模块由三部分组成：

jackson-core 核心包，提供基于”流模式”解析的相关 API，它包括 JsonPaser 和 JsonGenerator。Jackson 内部实现正是通过高性能的流模式 API 的 JsonGenerator 和 JsonParser 来生成和解析 json。

jackson-annotations 注解包，提供标准注解功能；

jackson-databind 数据绑定包，提供基于”对象绑定” 解析的相关 API（ ObjectMapper ）和”树模型” 解析的相关 API（JsonNode）；基于”对象绑定” 解析的 API 和”树模型”解析的 API 依赖基于”流模式”解析的 API。

为什么Jackson的介绍这么长啊？因为它也是本人的最爱。

## Json-lib

项目地址：http://json-lib.sourceforge.net/index.html

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，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。

JSON.JAR包虽然可以单独是用，但是在web项目中，尤其是在使用框架的时候，会涉及到类的映射，

这有时候回到JSON的使用产出影响，所以必须导入beanutil包，logging，

并且在一些框架中，大多使用基本数据类型的封装类，为了提高JSON的安全性，建议添加lang包，

此外，collections,以及ezmorph，也建议一同使用

### 简单使用

//1.创建对象

JSONObject json = new JSONObject();

//2.向JSON中存数据

json.put("name", JSONArray.fromObject(li));

//3.响应结果

response.getWriter().print(json.toString());

### JSONObject

JSONObject类本身是一个Map，所以学习它很方便。

JSONObject jo = new JSONObject();

jo.put("name", "zhangSan");

jo.put("age", "18");

jo.put("sex", "male");

System.out.println(jo.toString());[输出为：{"name":"zhangSan","age":"18","sex":"male"}]

Person person = new Person("liSi", 18, "female");

JSONObject jo = JSONObject.fromObject(person)[常用：把JavaBean对象转换成json];

System.out.println(jo.toString());

Map map = new HashMap();

map.put("name", "wangWu");

map.put("age", "81");

map.put("sex", "male");

JSONObject jo = JSONObject.fromObject(map)[把Map转换成JSON];

System.out.println(jo.toString());

String xml = "<person><name>zhaoLiu</name><age>59</age><sex>female</sex></person>";

XMLSerializer serial = new XMLSerializer();

JSONObject jo = (JSONObject)serial.read(xml);[把xml转换成JSON，这需要导入xom-1.1.jar]

System.out.println(jo.toString());

### JSONArray

JSONArray本身是一个List，所以使用起来很方便。

JSONArray ja = new JSONArray();

Person p1 = new Person("zhangSan", 18, "male");

Person p2 = new Person("liSi", 23, "female");

ja.add(p1);

ja.add(p2);

System.out.println(ja.toString());

Person p1 = new Person("zhangSan", 18, "male");

Person p2 = new Person("liSi", 23, "female");

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

list.add(p1);

list.add(p2);

JSONArray ja = JSONArray.fromObject(list)[常用：把list转换成JSONArray];

System.out.println(ja.toString());

Person p1 = new Person("zhangSan", 18, "male");

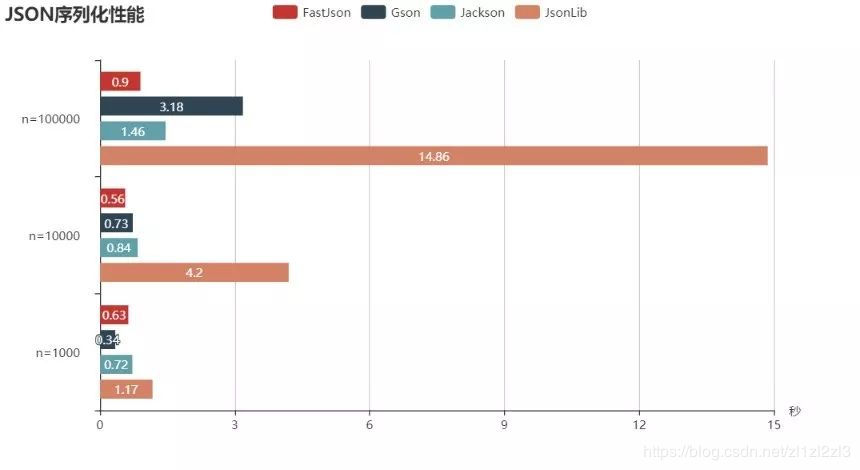
Person p2 = new Person("liSi", 23, "female");

Person[] persons = {p1, p2};

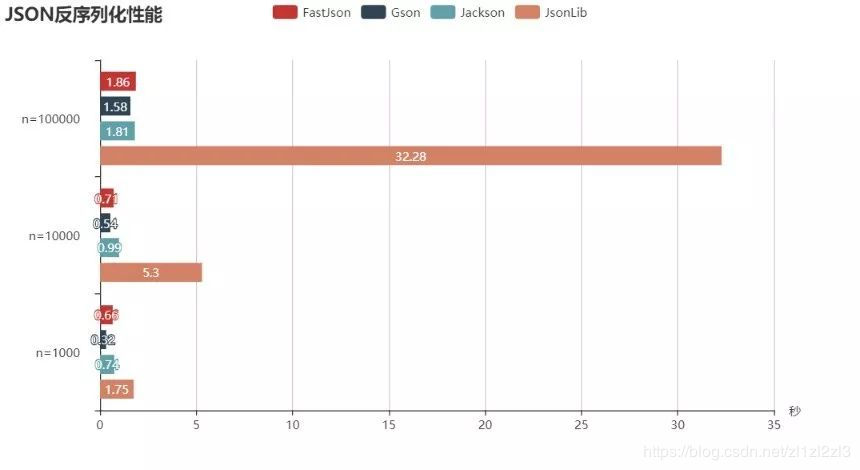
JSONArray ja = JSONArray.fromObject(persons)[把数组转换成JSONArray];

System.out.println(ja.toString());

# 性能测试



从上面的测试结果可以看出，序列化次数比较小的时候，Gson性能最好，当不断增加的时候到了100000，Gson明细弱于Jackson和FastJson， 这时候FastJson性能是真的牛，另外还可以看到不管数量少还是多，Jackson一直表现优异。而那个Json-lib简直就是来搞笑的。



反序列化的时候，Gson、Jackson和FastJson区别不大，性能都很优异，而那个Json-lib还是来继续搞笑的。