## 数据

### 数据格式

#### [JSON-lib框架，转换JSON、XML不再困难](http://www.cnblogs.com/hoojo/archive/2011/04/21/2023805.html)、

Json-lib可以将Java对象转成json格式的字符串，也可以将Java对象转换成xml格式的文档，同样可以将json字符串转换成Java对象或是将xml字符串转换成Java对象。

**一、 准备工作**

1、 首先要去官方下载json-lib工具包

下载地址：

<http://sourceforge.net/projects/json-lib/files/json-lib/json-lib-2.4/>

目前最新的是2.4的版本，本示例中使用的是v2.3；json-lib还需要以下依赖包：

jakarta commons-lang 2.5

jakarta commons-beanutils 1.8.0

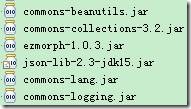
jakarta commons-collections 3.2.1

jakarta commons-logging 1.1.1

ezmorph 1.0.6

官方网址：<http://json-lib.sourceforge.net/>

然后在工程中添加如下jar包：

[](http://images.cnblogs.com/cnblogs_com/hoojo/201104/201104211723204815.jpg)

当然你也可以用2.4的json-lib库

你可以在这里看看官方提供的示例：

<http://json-lib.sourceforge.net/usage.html>

由于本次介绍的示例需要junit工具，所以你还需要添加junit的jar文件，版本是4.8.2版本的，下载地址：<https://github.com/KentBeck/junit/downloads>

如果你还不了解JSON是什么？那么你应该可以看看<http://www.json.org/json-zh.html>

#### [Jackson框架](http://blog.csdn.net/laixiaonian/article/details/8144920)

Jackson可以轻松的将Java对象转换成json对象和xml文档，同样也可以将json、xml转换成Java对象。

前面有介绍过json-lib这个框架，在线博文：<http://www.cnblogs.com/hoojo/archive/2011/04/21/2023805.html>

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。

**一、准备工作**

1、 下载依赖库jar包

Jackson的jar all下载地址：<http://jackson.codehaus.org/1.7.6/jackson-all-1.7.6.jar>

然后在工程中导入这个jar包即可开始工作

官方示例：<http://wiki.fasterxml.com/JacksonInFiveMinutes>

因为下面的程序是用junit测试用例运行的，所以还得添加junit的jar包。版本是junit-4.2.8

如果你需要转换xml，那么还需要stax2-api.jar

2、 测试类基本代码如下

package com.hoo.test;

import java.io.IOException;

import java.io.StringWriter;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Iterator;

import java.util.LinkedHashMap;

import java.util.List;

import java.util.Map;

import java.util.Set;

import org.codehaus.jackson.JsonEncoding;

import org.codehaus.jackson.JsonGenerationException;

import org.codehaus.jackson.JsonGenerator;

import org.codehaus.jackson.JsonParseException;

import org.codehaus.jackson.map.JsonMappingException;

import org.codehaus.jackson.map.ObjectMapper;

import org.codehaus.jackson.node.JsonNodeFactory;

import org.codehaus.jackson.xml.XmlMapper;

import org.junit.After;

import org.junit.Before;

import org.junit.Test;

import com.hoo.entity.AccountBean;

/\*\*

\* <b>function:</b>Jackson 将java对象转换成JSON字符串，也可以将JSON字符串转换成java对象

\* jar-lib-version: jackson-all-1.6.2

\* jettison-1.0.1

\* @author hoojo

\* @createDate 2010-11-23 下午04:54:53

\* @file JacksonTest.java

\* @package com.hoo.test

\* @project Spring3

\* @blog http://blog.csdn.net/IBM\_hoojo

\* @email hoojo\_@126.com

\* @version 1.0

\*/

@SuppressWarnings("unchecked")

public class JacksonTest {

private JsonGenerator jsonGenerator = null;

private ObjectMapper objectMapper = null;

private AccountBean bean = null;

@Before

public void init() {

bean = new AccountBean();

bean.setAddress("china-Guangzhou");

bean.setEmail("hoojo\_@126.com");

bean.setId(1);

bean.setName("hoojo");

objectMapper = new ObjectMapper();

try {

jsonGenerator = objectMapper.getJsonFactory().createJsonGenerator(System.out, JsonEncoding.UTF8);

} catch (IOException e) {

e.printStackTrace();

}

}

@After

public void destory() {

try {

if (jsonGenerator != null) {

jsonGenerator.flush();

}

if (!jsonGenerator.isClosed()) {

jsonGenerator.close();

}

jsonGenerator = null;

objectMapper = null;

bean = null;

System.gc();

} catch (IOException e) {

e.printStackTrace();

}

}

}

3、 所需要的JavaEntity

package com.hoo.entity;

public class AccountBean {

private int id;

private String name;

private String email;

private String address;

private Birthday birthday;

//getter、setter

@Override

public String toString() {

return this.name + "#" + this.id + "#" + this.address + "#" + this.birthday + "#" + this.email;

}

}

Birthday

package com.hoo.entity;

public class Birthday {

private String birthday;

public Birthday(String birthday) {

super();

this.birthday = birthday;

}

//getter、setter

public Birthday() {}

@Override

public String toString() {

return this.birthday;

}

}

**二、Java对象转换成JSON**

1、 JavaBean(Entity/Model)转换成JSON

/\*\*

\* <b>function:</b>将java对象转换成json字符串

\* @author hoojo

\* @createDate 2010-11-23 下午06:01:10

\*/

@Test

public void writeEntityJSON() {

try {

System.out.println("jsonGenerator");

//writeObject可以转换java对象，eg:JavaBean/Map/List/Array等

jsonGenerator.writeObject(bean);

System.out.println();

System.out.println("ObjectMapper");

//writeValue具有和writeObject相同的功能

objectMapper.writeValue(System.out, bean);

} catch (IOException e) {

e.printStackTrace();

}

}

运行后结果如下：

jsonGenerator

{"address":"china-Guangzhou","name":"hoojo","id":1,"birthday":null,"email":"hoojo\_@126.com"}

ObjectMapper

{"address":"china-Guangzhou","name":"hoojo","id":1,"birthday":null,"email":"hoojo\_@126.com"}

上面分别利用JsonGenerator的writeObject方法和ObjectMapper的writeValue方法完成对Java对象的转换，二者传递的参数及构造的方式不同；JsonGenerator的创建依赖于ObjectMapper对象。也就是说如果你要使用JsonGenerator来转换JSON，那么你必须创建一个ObjectMapper。但是你用ObjectMapper来转换JSON，则不需要JSONGenerator。

objectMapper的writeValue方法可以将一个Java对象转换成JSON。这个方法的参数一，需要提供一个输出流，转换后可以通过这个流来输出转换后的内容。或是提供一个File，将转换后的内容写入到File中。当然，这个参数也可以接收一个JSONGenerator，然后通过JSONGenerator来输出转换后的信息。第二个参数是将要被转换的Java对象。如果用三个参数的方法，那么是一个Config。这个config可以提供一些转换时的规则，过指定的Java对象的某些属性进行过滤或转换等。

2、 将Map集合转换成Json字符串

/\*\*

\* <b>function:</b>将map转换成json字符串

\* @author hoojo

\* @createDate 2010-11-23 下午06:05:26

\*/

@Test

public void writeMapJSON() {

try {

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

map.put("name", bean.getName());

map.put("account", bean);

bean = new AccountBean();

bean.setAddress("china-Beijin");

bean.setEmail("hoojo@qq.com");

map.put("account2", bean);

System.out.println("jsonGenerator");

jsonGenerator.writeObject(map);

System.out.println("");

System.out.println("objectMapper");

objectMapper.writeValue(System.out, map);

} catch (IOException e) {

e.printStackTrace();

}

}

转换后结果如下：

jsonGenerator

{"account2":{"address":"china-Beijin","name":null,"id":0,"birthday":null,"email":"hoojo@qq.com"},"name":"hoojo",

"account":{"address":"china-Guangzhou","name":"hoojo","id":1,"birthday":null,"email":"hoojo\_@126.com"}}

objectMapper

{"account2":{"address":"china-Beijin","name":null,"id":0,"birthday":null,"email":"hoojo@qq.com"},"name":"hoojo",

"account":{"address":"china-Guangzhou","name":"hoojo","id":1,"birthday":null,"email":"hoojo\_@126.com"}}

3、 将List集合转换成json

/\*\*

\* <b>function:</b>将list集合转换成json字符串

\* @author hoojo

\* @createDate 2010-11-23 下午06:05:59

\*/

@Test

public void writeListJSON() {

try {

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

list.add(bean);

bean = new AccountBean();

bean.setId(2);

bean.setAddress("address2");

bean.setEmail("email2");

bean.setName("haha2");

list.add(bean);

System.out.println("jsonGenerator");

//list转换成JSON字符串

jsonGenerator.writeObject(list);

System.out.println();

System.out.println("ObjectMapper");

//用objectMapper直接返回list转换成的JSON字符串

System.out.println("1###" + objectMapper.writeValueAsString(list));

System.out.print("2###");

//objectMapper list转换成JSON字符串

objectMapper.writeValue(System.out, list);

} catch (IOException e) {

e.printStackTrace();

}

}

结果如下：

jsonGenerator

[{"address":"china-Guangzhou","name":"hoojo","id":1,"birthday":null,"email":"hoojo\_@126.com"},

{"address":"address2","name":"haha2","id":2,"birthday":null,"email":"email2"}]

ObjectMapper

1###[{"address":"china-Guangzhou","name":"hoojo","id":1,"birthday":null,"email":"hoojo\_@126.com"},

{"address":"address2","name":"haha2","id":2,"birthday":null,"email":"email2"}]

2###[{"address":"china-Guangzhou","name":"hoojo","id":1,"birthday":null,"email":"hoojo\_@126.com"},

{"address":"address2","name":"haha2","id":2,"birthday":null,"email":"email2"}]

外面就是多了个[]中括号；同样Array也可以转换，转换的JSON和上面的结果是一样的，这里就不再转换了。~.~

4、下面来看看jackson提供的一些类型，用这些类型完成json转换；如果你使用这些类型转换JSON的话，那么你即使没有JavaBean(Entity)也可以完成复杂的Java类型的JSON转换。下面用到这些类型构建一个复杂的Java对象，并完成JSON转换。

@Test

public void writeOthersJSON() {

try {

String[] arr = { "a", "b", "c" };

System.out.println("jsonGenerator");

String str = "hello world jackson!";

//byte

jsonGenerator.writeBinary(str.getBytes());

//boolean

jsonGenerator.writeBoolean(true);

//null

jsonGenerator.writeNull();

//float

jsonGenerator.writeNumber(2.2f);

//char

jsonGenerator.writeRaw("c");

//String

jsonGenerator.writeRaw(str, 5, 10);

//String

jsonGenerator.writeRawValue(str, 5, 5);

//String

jsonGenerator.writeString(str);

jsonGenerator.writeTree(JsonNodeFactory.instance.POJONode(str));

System.out.println();

//Object

jsonGenerator.writeStartObject();//{

jsonGenerator.writeObjectFieldStart("user");//user:{

jsonGenerator.writeStringField("name", "jackson");//name:jackson

jsonGenerator.writeBooleanField("sex", true);//sex:true

jsonGenerator.writeNumberField("age", 22);//age:22

jsonGenerator.writeEndObject();//}

jsonGenerator.writeArrayFieldStart("infos");//infos:[

jsonGenerator.writeNumber(22);//22

jsonGenerator.writeString("this is array");//this is array

jsonGenerator.writeEndArray();//]

jsonGenerator.writeEndObject();//}

AccountBean bean = new AccountBean();

bean.setAddress("address");

bean.setEmail("email");

bean.setId(1);

bean.setName("haha");

//complex Object

jsonGenerator.writeStartObject();//{

jsonGenerator.writeObjectField("user", bean);//user:{bean}

jsonGenerator.writeObjectField("infos", arr);//infos:[array]

jsonGenerator.writeEndObject();//}

} catch (Exception e) {

e.printStackTrace();

}

}

运行后，结果如下：

jsonGenerator

"aGVsbG8gd29ybGQgamFja3NvbiE=" true null 2.2c world jac worl "hello world jackson!" "hello world jackson!"

{"user":{"name":"jackson","sex":true,"age":22},"infos":[22,"this is array"]}

{"user":{"address":"address","name":"haha","id":1,"birthday":null,"email":"email"},"infos":["a","b","c"]}

怎么样？构造的json字符串和输出的结果是一致的吧。关键看懂用JSONGenerator提供的方法，完成一个Object的构建。

**三、JSON转换成Java对象**

1、 将json字符串转换成JavaBean对象

@Test

public void readJson2Entity() {

String json = "{\"address\":\"address\",\"name\":\"haha\",\"id\":1,\"email\":\"email\"}";

try {

AccountBean acc = objectMapper.readValue(json, AccountBean.class);

System.out.println(acc.getName());

System.out.println(acc);

} catch (JsonParseException e) {

e.printStackTrace();

} catch (JsonMappingException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

}

很简单，用到了ObjectMapper这个对象的readValue这个方法，这个方法需要提供2个参数。第一个参数就是解析的JSON字符串，第二个参数是即将将这个JSON解析吃什么Java对象，Java对象的类型。当然，还有其他相同签名方法，如果你有兴趣可以一一尝试使用方法，当然使用的方法和当前使用的方法大同小异。运行后，结果如下：

haha

haha#1#address#null#email

2、 将json字符串转换成List<Map>集合

/\*\*

\* <b>function:</b>json字符串转换成list<map>

\* @author hoojo

\* @createDate 2010-11-23 下午06:12:01

\*/

@Test

public void readJson2List() {

String json = "[{\"address\": \"address2\",\"name\":\"haha2\",\"id\":2,\"email\":\"email2\"},"+

"{\"address\":\"address\",\"name\":\"haha\",\"id\":1,\"email\":\"email\"}]";

try {

List<LinkedHashMap<String, Object>> list = objectMapper.readValue(json, List.class);

System.out.println(list.size());

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

Map<String, Object> map = list.get(i);

Set<String> set = map.keySet();

for (Iterator<String> it = set.iterator();it.hasNext();) {

String key = it.next();

System.out.println(key + ":" + map.get(key));

}

}

} catch (JsonParseException e) {

e.printStackTrace();

} catch (JsonMappingException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

}

尝试过将上面的JSON转换成List，然后List中存放AccountBean，但结果失败了。但是支持Map集合。因为你转成List.class，但是不知道List存放何种类型。只好默然Map类型。因为所有的对象都可以转换成Map结合，运行后结果如下：

2

address:address2

name:haha2

id:2

email:email2

address:address

name:haha

id:1

email:email

3、 Json字符串转换成Array数组，由于上面的泛型转换不能识别到集合中的对象类型。所有这里用对象数组，可以解决这个问题。只不过它不再是集合，而是一个数组。当然这个不重要，你可以用Arrays.asList将其转换成List即可。

/\*\*

\* <b>function:</b>json字符串转换成Array

\* @author hoojo

\* @createDate 2010-11-23 下午06:14:01

\*/

@Test

public void readJson2Array() {

String json = "[{\"address\": \"address2\",\"name\":\"haha2\",\"id\":2,\"email\":\"email2\"},"+

"{\"address\":\"address\",\"name\":\"haha\",\"id\":1,\"email\":\"email\"}]";

try {

AccountBean[] arr = objectMapper.readValue(json, AccountBean[].class);

System.out.println(arr.length);

for (int i = 0; i < arr.length; i++) {

System.out.println(arr[i]);

}

} catch (JsonParseException e) {

e.printStackTrace();

} catch (JsonMappingException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

}

运行后的结果：

2

haha2#2#address2#null#email2

haha#1#address#null#email

4、 Json字符串转换成Map集合

/\*\*

\* <b>function:</b>json字符串转换Map集合

\* @author hoojo

\* @createDate Nov 27, 2010 3:00:06 PM

\*/

@Test

public void readJson2Map() {

String json = "{\"success\":true,\"A\":{\"address\": \"address2\",\"name\":\"haha2\",\"id\":2,\"email\":\"email2\"},"+

"\"B\":{\"address\":\"address\",\"name\":\"haha\",\"id\":1,\"email\":\"email\"}}";

try {

Map<String, Map<String, Object>> maps = objectMapper.readValue(json, Map.class);

System.out.println(maps.size());

Set<String> key = maps.keySet();

Iterator<String> iter = key.iterator();

while (iter.hasNext()) {

String field = iter.next();

System.out.println(field + ":" + maps.get(field));

}

} catch (JsonParseException e) {

e.printStackTrace();

} catch (JsonMappingException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

}

运行后结果如下：

3

success:true

A:{address=address2, name=haha2, id=2, email=email2}

B:{address=address, name=haha, id=1, email=email}

**四、Jackson对XML的支持**

Jackson也可以完成java对象到xml的转换，转换后的结果要比json-lib更直观，不过它依赖于stax2-api.jar这个jar包。

/\*\*

\* <b>function:</b>java对象转换成xml文档

\* 需要额外的jar包 stax2-api.jar

\* @author hoojo

\* @createDate 2010-11-23 下午06:11:21

\*/

@Test

public void writeObject2Xml() {

//stax2-api-3.0.2.jar

System.out.println("XmlMapper");

XmlMapper xml = new XmlMapper();

try {

//javaBean转换成xml

//xml.writeValue(System.out, bean);

StringWriter sw = new StringWriter();

xml.writeValue(sw, bean);

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

//List转换成xml

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

list.add(bean);

list.add(bean);

System.out.println(xml.writeValueAsString(list));

//Map转换xml文档

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

map.put("A", bean);

map.put("B", bean);

System.out.println(xml.writeValueAsString(map));

} catch (JsonGenerationException e) {

e.printStackTrace();

} catch (JsonMappingException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

}

运行上面的方法，结果如下：

XmlMapper

<unknown><address>china-Guangzhou</address><name>hoojo</name><id>1</id><birthday/><email>hoojo\_@126.com</email></unknown>

<unknown><unknown><address>china-Guangzhou</address><name>hoojo</name><id>1</id><birthday/><email>hoojo\_@126.com</email></unknown>

<email><address>china-Guangzhou</address><name>hoojo</name><id>1</id><birthday/><email>hoojo\_@126.com</email></email></unknown>

<unknown><A><address>china-Guangzhou</address><name>hoojo</name><id>1</id><birthday/><email>hoojo\_@126.com</email></A>

<B><address>china-Guangzhou</address><name>hoojo</name><id>1</id><birthday/><email>hoojo\_@126.com</email></B></unknown>

看结果，根节点都是unknown 这个问题还没有解决，由于根节点没有转换出来，所有导致解析xml到Java对象，也无法完成。

一个在线获取google天气的json解析例子；

**[java]** [view plaincopyprint?](http://blog.csdn.net/laixiaonian/article/details/8144920)

1. package com.lai.chap7.weather;
2. import java.io.BufferedReader;
3. import java.io.IOException;
4. import java.io.InputStreamReader;
5. import java.net.HttpURLConnection;
6. import java.net.URL;
7. import java.nio.charset.Charset;
8. public class GetWeatherFromGoogle {
9. /\*\*
10. \* @param args
11. \*/
12. public static void main(String[] args) {
13. GetWeatherFromGoogle getWeatherFromGoogle= new GetWeatherFromGoogle();
14. System.out.println(getWeatherFromGoogle.gestInfro());
15. }
16. public String gestInfro(){
17. {
18. String allOrg="";
19. HttpURLConnection connection =null;
20. try{
21. String hrmsWebservicesOrgURL ="http://weather.china.xappengine.com/api?city=shenzhen";
22. URL getUrl = new URL(hrmsWebservicesOrgURL);
23. connection = (HttpURLConnection) getUrl.openConnection();
24. connection.connect();
25. BufferedReader reader = new BufferedReader(new InputStreamReader(
26. connection.getInputStream(),Charset.forName("UTF-8")));
27. StringBuffer temp = new StringBuffer();
28. String lines = reader.readLine();
29. while (lines!= null) {
30. temp.append(lines);
31. lines = reader.readLine();
32. }
33. reader.close();
34. allOrg = temp.toString();
35. }catch(IOException io){
36. io.printStackTrace();
37. }catch(Exception e){
38. e.printStackTrace();
39. }finally{
40. connection.disconnect();
41. }
42. return allOrg;
43. }
44. }
45. }

package com.lai.chap7.weather;

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.net.HttpURLConnection;

import java.net.URL;

import java.nio.charset.Charset;

public class GetWeatherFromGoogle {

/\*\*

\* @param args

\*/

public static void main(String[] args) {

GetWeatherFromGoogle getWeatherFromGoogle= new GetWeatherFromGoogle();

System.out.println(getWeatherFromGoogle.gestInfro());

}

public String gestInfro(){

{

String allOrg="";

HttpURLConnection connection =null;

try{

String hrmsWebservicesOrgURL ="http://weather.china.xappengine.com/api?city=shenzhen";

URL getUrl = new URL(hrmsWebservicesOrgURL);

connection = (HttpURLConnection) getUrl.openConnection();

connection.connect();

BufferedReader reader = new BufferedReader(new InputStreamReader(

connection.getInputStream(),Charset.forName("UTF-8")));

StringBuffer temp = new StringBuffer();

String lines = reader.readLine();

while (lines!= null) {

temp.append(lines);

lines = reader.readLine();

}

reader.close();

allOrg = temp.toString();

}catch(IOException io){

io.printStackTrace();

}catch(Exception e){

e.printStackTrace();

}finally{

connection.disconnect();

}

return allOrg;

}

}

}

**[java]** [view plaincopyprint?](http://blog.csdn.net/laixiaonian/article/details/8144920)

1. package com.lai.chap7.weather;
2. import java.io.IOException;
3. import java.util.Iterator;
4. import java.util.Map;
5. import java.util.Set;
6. import org.codehaus.jackson.JsonParseException;
7. import org.codehaus.jackson.map.JsonMappingException;
8. import org.codehaus.jackson.map.ObjectMapper;
9. public class ParserJson {
10. /\*\*
11. \* @param args
12. \*/
13. public static void main(String[] args) {
14. GetWeatherFromGoogle getWeatherFromGoogle= new GetWeatherFromGoogle();
15. // System.out.println(getWeatherFromGoogle.gestInfro());
16. String s = getWeatherFromGoogle.gestInfro();
17. ParserJson parserJson=new ParserJson();
18. parserJson.readJson2Map(s);
19. }
20. public static void readJson2Map(String json) {
21. try {
22. ObjectMapper objectMapper = objectMapper = new ObjectMapper();
23. Map<String, Map<String, Object>> maps = objectMapper.readValue(json, Map.class);
24. System.out.println(maps.size());
25. Set<String> key = maps.keySet();
26. Iterator<String> iter = key.iterator();
27. while (iter.hasNext()) {
28. String field = iter.next();
29. System.out.println(field + ":" + maps.get(field));
30. }
31. } catch (JsonParseException e) {
32. e.printStackTrace();
33. } catch (JsonMappingException e) {
34. e.printStackTrace();
35. } catch (IOException e) {
36. e.printStackTrace();
37. }
38. }
39. }

package com.lai.chap7.weather;

import java.io.IOException;

import java.util.Iterator;

import java.util.Map;

import java.util.Set;

import org.codehaus.jackson.JsonParseException;

import org.codehaus.jackson.map.JsonMappingException;

import org.codehaus.jackson.map.ObjectMapper;

public class ParserJson {

/\*\*

\* @param args

\*/

public static void main(String[] args) {

GetWeatherFromGoogle getWeatherFromGoogle= new GetWeatherFromGoogle();

// System.out.println(getWeatherFromGoogle.gestInfro());

String s = getWeatherFromGoogle.gestInfro();

ParserJson parserJson=new ParserJson();

parserJson.readJson2Map(s);

}

public static void readJson2Map(String json) {

try {

ObjectMapper objectMapper = objectMapper = new ObjectMapper();

Map<String, Map<String, Object>> maps = objectMapper.readValue(json, Map.class);

System.out.println(maps.size());

Set<String> key = maps.keySet();

Iterator<String> iter = key.iterator();

while (iter.hasNext()) {

String field = iter.next();

System.out.println(field + ":" + maps.get(field));

}

} catch (JsonParseException e) {

e.printStackTrace();

} catch (JsonMappingException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

}

}