五分钟Jackson入门（一） JSON数据与Java对象相互转换（附项目源码）

码农场 > 编程开发 > Java　2013-12-31　阅读(10228)　评论(4)

JSON（**J**ava**S**cript **O**bject **N**otation）是一种轻量级的数据交换语言，以文字为基础，不仅便于机器解析，而且易于让人阅读。

谈起数据持久化储存，Java和MFC有序列化，Windows下还经常使用INI和注册表，甚至可以自己scanf printf一套新的标准。但是在网络上传递数据，尤其是需要时不时地检查调试数据的时候，则需要一种人眼可阅读的语言，这时就轮到JSON上场了。

serialize化的数据直接用文本工具打开看到的都是乱码，不同于serialize，JSON数据直接可以打开阅读，并且几乎不需要学习新的语法，非常便于阅读。比如：

```
{
  "name" : { "first" : "Joe", "last" : "Sixpack" },
  "gender" : "MALE",
  "verified" : false,
  "userImage" : "Rm9vYmFyIQ=="
}
```

一眼可以看出这段 JSON data 描述了一个用户的基本信息，其中name字段对应的是另一个对象（凡是大括号包起来的就是一个对象）。

那么在Java里如何解析和生成JSON数据呢？

Jackson 框架可以很轻松地完成这一任务：http://jackson.codehaus.org/

下面来看看如何将上述JSON数据转化为Java的对象。

首先去下载Jackson的库：

下载地址：http://wiki.fasterxml.com/JacksonDownload

Jackson 框架2.x似乎分成了三个jar包：

> Core (release notes)
>
> Annotations (release notes)
>
> Databind (release notes)

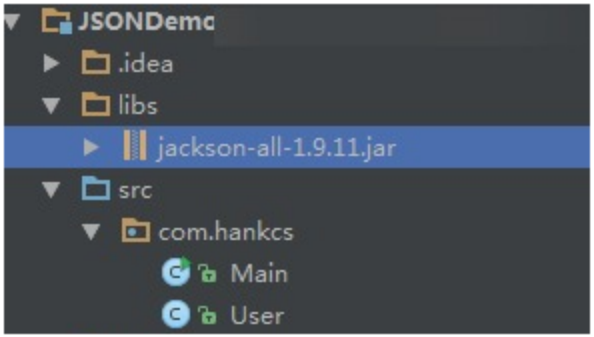
第一个是核心jar，其余的是拓展。

而Jackson 框架1.x则集成为一整个jar包，非常便携，没有其他要求的话就用这个好了，1.x最新的版本下载地址：

http://jackson.codehaus.org/1.9.11/jackson-all-1.9.11.jar

然后建立测试项目，导入jar

项目结构：



其中User类是与上述JSON数据配套的Java类：

```java
package com.hankcs;

import java.util.Arrays;

/**
 * @author Hankcs
 */
public class User
{
    public enum Gender
    {
        MALE, FEMALE
    };

    public static class Name
    {
        private String _first, _last;

        public String getFirst()
        {
            return _first;
        }

        public String getLast()
        {
            return _last;
        }

        public void setFirst(String s)
        {
            _first = s;
```

```
32.         }
33.
34.         public void setLast(String s)
35.         {
36.             _last = s;
37.         }
38.
39.         @Override
40.         public String toString()
41.         {
42.             return "Name{" +
43.                     "_first='" + _first + '\'' +
44.                     ", _last='" + _last + '\'' +
45.                     '}';
46.         }
47.     }
48.
49.     private Gender _gender;
50.     private Name _name;
51.     private boolean _isVerified;
52.     private byte[] _userImage;
53.
54.     public Name getName()
55.     {
56.         return _name;
57.     }
58.
59.     public boolean isVerified()
60.     {
61.         return _isVerified;
62.     }
63.
64.     public Gender getGender()
65.     {
66.         return _gender;
67.     }
68.
69.     public byte[] getUserImage()
```

```
    {
        return _userImage;
    }

    public void setName(Name n)
    {
        _name = n;
    }

    public void setVerified(boolean b)
    {
        _isVerified = b;
    }

    public void setGender(Gender g)
    {
        _gender = g;
    }

    public void setUserImage(byte[] b)
    {
        _userImage = b;
    }

    @Override
    public String toString()
    {
        return "User{" +
                "_gender=" + _gender +
                ", _name=" + _name +
                ", _isVerified=" + _isVerified +
                ", _userImage=" + Arrays.toString(_userImage) +
                '}';
    }
}
```

/user.json就是等待解析JSON数据。

Main类：

```
package com.hankcs;

import org.codehaus.jackson.map.ObjectMapper;

import java.io.File;
import java.io.IOException;

public class Main
{

    public static void main(String[] args)
    {
        // write your code here
        ObjectMapper mapper = new ObjectMapper(); // can reuse, share globally
        try
        {
            // 尝试从JSON中读取对象
            User user = mapper.readValue(new File("user.json"), User.class);
            System.out.println(user);
            user.setGender(User.Gender.FEMALE);
            mapper.writeValue(new File("user-modified.json"), user);
        } catch (IOException e)
        {
            e.printStackTrace();
        }
    }
}
```

只需要两句话就能解析出来：

```
ObjectMapper mapper = new ObjectMapper();
User user = mapper.readValue(new File("user.json"), User.class);
```

写入也很简单，一句话搞定：

```
1.  mapper.writeValue(new File("user-modified.json"), user);
```

输出：

```
1.  User{_gender=MALE, _name=Name{_first='Joe', _last='Sixpack'}, _isVerified=false, _userImage=[70, 111,
```

项目源码：http://pan.baidu.com/s/1dDiQ8Vn

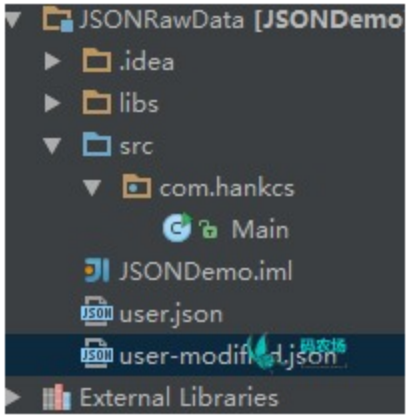
参考资料：http://wiki.fasterxml.com/JacksonInFiveMinutes#Full_Data_Binding_.28POJO.29_Example

五分钟Jackson入门（二）JSON数据与Map数据相互转换（附项目源码）

 2013-12-31

上篇入门教程里实现了JSON数据与Java对象的相互转换，在那篇文章里，我们编写了Java对象的class所以才能存放它。实际上，在不需要class的场景下，一个Map就可以简单地将对象表示出来。

还是拿上次的Demo项目做演示，这次我们删除User.class，项目结构：



Main里改成：

```
1.  package com.hankcs;
2.
3.  import org.codehaus.jackson.map.ObjectMapper;
```

```java

import java.io.File;
import java.io.IOException;
import java.util.HashMap;
import java.util.Map;

public class Main
{

    public static void main(String[] args)
    {
        // write your code here
        ObjectMapper mapper = new ObjectMapper(); // can reuse, share globally
        try
        {
            // 读取JSON数据
            Map<String,Object> userData = mapper.readValue(new File("user.json"), Map.class);
            System.out.println(userData);
            // 写入JSON数据
            userData = new HashMap<String,Object>();
            Map<String,String> nameStruct = new HashMap<String,String>();
            nameStruct.put("first", "Joe");
            nameStruct.put("last", "Hankcs");
            userData.put("name", nameStruct);
            userData.put("gender", "MALE");
            userData.put("verified", Boolean.FALSE);
            userData.put("userImage", "Rm9vYmFyIQ==");
            mapper.writeValue(new File("user-modified.json"), userData);
        } catch (IOException e)
        {
            e.printStackTrace();
        }
    }
}
```

输出：

```
{name={first=Joe, last=Sixpack}, gender=MALE, verified=false, userImage=Rm9vYmFyIQ==}
```

/user-modified.json：

```
{"verified":false,"name":{"last":"Hankcs","first":"Joe"},"userImage":"Rm9vYmFyIQ==","gender":"MALE"}
```

值得注意的是name字段的处理，name字段是一个嵌套的map，在JSON数据里，每个Map都被一个括号包起来了。

估计Jackson库使用了反射来处理这些字段，所以所有的字段必须是对象或者包装类。一个官方的对照表：

JSON Type	Java Type
object	LinkedHashMap<String, Object>
array	ArrayList<Object>
string	String
number (no fraction)	Integer, Long or BigInteger (smallest applicable)
number (fraction)	Double (configurable to use BigDecimal)
true|false	Boolean
null	null

上文的Map对象类型是string-object类型的，假如遇上了泛型会怎样呢？比如Map<String,User>，要知道，Java里的擦除机制可不允许使用Map<String,User>.class的。

Jackson框架考虑到了这一点，这要这么干就行了：

```
Map<String,User> result = mapper.readValue(src, new TypeReference<Map<String,User>>() { });
```

关于这里的匿名类的作用，可以参考我写的获取T.class的一篇文章：

http://www.hankcs.com/program/t-class.html

项目源码：http://pan.baidu.com/s/1jGFk04I

参考资料：http://wiki.fasterxml.com/JacksonInFiveMinutes#Full_Data_Binding_.28POJO.29_Example

五分钟Jackson入门（三） JSON数据类XML转换（附项目源码）

 2013-12-31

JSON数据看起来就像一棵树，也可以用类似于XML的解析方法来解析。将上篇文章的Demo改为：

```java
package com.hankcs;

import org.codehaus.jackson.JsonNode;
import org.codehaus.jackson.map.ObjectMapper;
import org.codehaus.jackson.node.ObjectNode;

import java.io.File;
import java.io.IOException;
import java.util.HashMap;
import java.util.Map;

public class Main
{

    public static void main(String[] args) throws IOException
    {
        ObjectMapper m = new ObjectMapper();
        // can either use mapper.readTree(source), or mapper.readValue(source, JsonNode.class);
        JsonNode rootNode = m.readTree(new File("user.json"));
        // ensure that "last name" isn't "Xmler"; if is, change to "Jsoner"
        JsonNode nameNode = rootNode.path("name");
        String lastName = nameNode.path("last").getTextValue();
        System.out.println(lastName);
        if ("xmler".equalsIgnoreCase(lastName))
        {
            ((ObjectNode) nameNode).put("last", "Jsoner");
        }
        // and write it out:
        m.writeValue(new File("user-modified.json"), rootNode);
    }
}
```

项目源码：http://pan.baidu.com/s/1iqFYe

参考：http://wiki.fasterxml.com/JacksonInFiveMinutes#Full_Data_Binding_.28POJO.29_Example

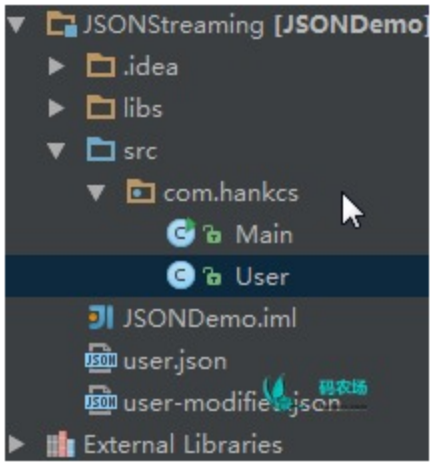
五分钟Jackson入门（四）JSON Streaming API（附项目源码）

 2013-12-31

据说Streaming API 的效率是最高的，写入的时候直接调用JsonGenerator.writexxxfield，最后一个close就flush到文件了。不过读取的时候则比较蠢（？），需要一个while循环，不断地将文件里的字段与对象的字段作是否等于的比较，然后setField。

这次Demo里需要用到User.class，在第一个Demo的基础上改好了：

项目结构：




```
1.  package com.hankcs;
2.
3.  import org.codehaus.jackson.*;
```

```
import org.codehaus.jackson.map.ObjectMapper;
import org.codehaus.jackson.node.ObjectNode;

import java.io.File;
import java.io.IOException;
import java.util.HashMap;
import java.util.Map;

public class Main
{

    public static void main(String[] args) throws IOException
    {
        JsonFactory f = new JsonFactory();
        JsonGenerator g = f.createJsonGenerator(new File("user.json"), JsonEncoding.UTF8);

        g.writeStartObject();
        g.writeObjectFieldStart("name");
        g.writeStringField("first", "Joe");
        g.writeStringField("last", "Hankcs");
        g.writeEndObject(); // for field 'name'
        g.writeStringField("gender", "MALE");
        g.writeBooleanField("verified", false);
        g.writeFieldName("userImage"); // no 'writeBinaryField' (yet?)
        byte[] binaryData = {0x1, 0x2, 0x3};
        g.writeBinary(binaryData);
        g.writeEndObject();
        g.close(); // important: will force flushing of output, close underlying output stream

        // 解析
//        JsonFactory f = new JsonFactory();
        JsonParser jp = f.createJsonParser(new File("user.json"));
        User user = new User();
        jp.nextToken(); // will return JsonToken.START_OBJECT (verify?)
        while (jp.nextToken() != JsonToken.END_OBJECT)
        {
            String fieldname = jp.getCurrentName();
            jp.nextToken(); // move to value, or START_OBJECT/START_ARRAY
```

```
42.             if ("name".equals(fieldname))
43.             { // contains an object
44.                 User.Name name = new User.Name();
45.                 while (jp.nextToken() != JsonToken.END_OBJECT)
46.                 {
47.                     String namefield = jp.getCurrentName();
48.                     jp.nextToken(); // move to value
49.                     if ("first".equals(namefield))
50.                     {
51.                         name.setFirst(jp.getText());
52.                     }
53.                     else if ("last".equals(namefield))
54.                     {
55.                         name.setLast(jp.getText());
56.                     }
57.                     else
58.                     {
59.                         throw new IllegalStateException("Unrecognized field '" + fieldname + "'!");
60.                     }
61.                 }
62.                 user.setName(name);
63.             }
64.             else if ("gender".equals(fieldname))
65.             {
66.                 user.setGender(User.Gender.valueOf(jp.getText()));
67.             }
68.             else if ("verified".equals(fieldname))
69.             {
70.                 user.setVerified(jp.getCurrentToken() == JsonToken.VALUE_TRUE);
71.             }
72.             else if ("userImage".equals(fieldname))
73.             {
74.                 user.setUserImage(jp.getBinaryValue());
75.             }
76.             else
77.             {
78.                 throw new IllegalStateException("Unrecognized field '" + fieldname + "'!");
```

```
            }
        }
        jp.close(); // ensure resources get cleaned up timely and properly
        System.out.println(user);
    }
}
```

输出：

```
User{_gender=MALE, _name=Name{_first='Joe', _last='Hankcs'}, _isVerified=false, _userImage=[1, 2, 3]}
```

对于数组类型的JSON数据，则简单了不少：

```
// 对于数组有更好的方法
        ObjectMapper mapper = new ObjectMapper();
        String json = "[{\"foo\": \"bar\"},{\"foo\": \"biz\"}]";
        jp = f.createJsonParser(json);
        // advance stream to START_ARRAY first:
        jp.nextToken();
        // and then each time, advance to opening START_OBJECT
        while (jp.nextToken() == JsonToken.START_OBJECT)
        {
            Foo foobar = mapper.readValue(jp, Foo.class);
            // process
            System.out.println(foobar);
            // after binding, stream points to closing END_OBJECT
        }
```

输出：

```
Foo{foo='bar'}
Foo{foo='biz'}
```

项目源码：http://pan.baidu.com/s/1nt8sFDB

参考：http://wiki.fasterxml.com/JacksonInFiveMinutes#Full_Data_Binding_.28POJO.29_Example

Jackson 框架，轻易转换JSON

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;
```

```java
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。