## json lib 简介

|  |
| --- |
| java对象生成JSON对象  输入: java对象  输出: json字符串 |

## 在maven中的依赖配置

|  |
| --- |
| <!-- json lib-->  <dependency>  <groupId>net.sf.json-lib</groupId>  <artifactId>json-lib</artifactId>  <version>2.1</version>  <classifier>jdk15</classifier>  </dependency> |

## JSONArray (输出JSON数组)

|  |
| --- |
| JSONArray jsonArray = **new** JSONArray();  jsonArray.add("xiaoyu");  jsonArray.add("wugang");  jsonArray.add("luochuan");  System.*out*.println(jsonArray.toString()); |
| 输出:  ["xiaoyu","wugang","luochuan"] |

## JSONObject (输出JSON对象)

|  |
| --- |
| JSONObject jsonObj = **new** JSONObject();  jsonObj.put("name", "jinwanlin");  jsonObj.put("age", "23"); |
| 输出:  {"name":"xiaoyu","age":"23"} |

## String转JSON

|  |
| --- |
| JSONObject shoppingObj = JSONObject.*fromObject*(shoppingListStr); |

## 从JSON对象中获取key为XX的值

|  |
| --- |
| if (infoObj.containsKey("tiny")) {  avatar = infoObj.getString("middle");  } |

**json-lib**

下载

到http://json-lib.sourceforge.net 下载相应版本的json-lib-x。

依据对jdk 的匹配，Json-lib有两个版本。

1. x-jdk13匹配于JDK 1.3.1及更高版本。

2. x-jdk15匹配于JDK 1.5，包括在JSONArray 和JSONObject中对Enums 的支持。 运行json还需要的包有（可检查最新版本，一般最新版本会向下兼容）：

\* jakarta commons-lang 2.4

\* jakarta commons-beanutils 1.7.0

\* jakarta commons-collections 3.2.1

\* jakarta commons-logging 1.1.1

\* ezmorph 1.0.4

如果工作于XML，则需要依赖于 xom1.1（官网打不开，找了好久都没有下到）

\* l 转化数组和集合

boolean[] boolArray = new boolean[]{true,false,true};

JSONArray jsonArray = JSONArray.fromObject(boolArray);

System.out.println(jsonArray);

输出：[true,false,true]

List list = new ArrayList();

list.add(“第一个”);

list.add(“第二个”);

JSONArray jsonArray = JSONArray.fromObject(list);

System.out.println(jsonArray);

输出：[“第一个”, “第二个”]

JSONArray jsonArray3 = JSONArray.fromObject("['json','is','easy']");

System.out.println(jsonArray3);

输出：[“json”, “is”, “'easy'”]

l 转化对象

转化Map

Map map = new HashMap();

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

map.put("bool",Boolean.TRUE);

map.put("int",new Integer(1));

map.put("arr",new String[]{"a","b"});

map.put("func","function(i){return this.arr[i];}");

JSONObject json = JSONObject.fromObject(map);

System.out.println(json);

输出：[“name”: “json”, “bool”:true, “int”,1, “arr”:[ “a”, “b”], “func”:function(i){return this.arr[i];}]

转化Bean

MyBean.java

public class MyBean {

private String name = "json";

private int pojoId = 1;

private String func1 = "function(i){return this.options[i]}";

private JSONFunction func2 = new JSONFunction(new String[]{"i"},"return this.options[i];");

//以下为get、set方法

}

MyBean bean = new MyBean();

JSONObject jsonObject = JSONObject.fromObject(bean);

System.out.println(jsonObject);

输出：

{"func1":function(i){return this.options[i]},"pojoId":1,"name":"json","func2":function(i){ return this.options[i]; }

从JSON到Beans

//转化为动态bean

String myjson = "{name=\"json\",bool:true,int:1,double:2.2,function:function(a){return a;},array:[1,2]}";

JSONObject json1 = JSONObject.fromString(myjson);

Object bean1 = JSONObject.toBean(json1);

由JSON生成XML

JSONObject json = new JSONObject(true);

XMLSerializer xmlSerializer = new XMLSerializer();

String xml = xmlSerializer.write(json);

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

输出：xml:<?xml version="1.0" encoding="UTF-8"?>

<o null="true"/>

JSONObject json2 = JSONObject.fromObject("{\"name\":\"json\",\"bool\":true,\"int\":1}");

String xml2 = xmlSerializer.write(json2);

System.out.println("xml2:" + xml2);

输出：xml2:<?xml version="1.0" encoding="UTF-8"?>

<o><bool type="boolean">true</bool><int type="number">1</int><name type="string">json</name></o>

JSONArray json3 = JSONArray.fromObject("[1,2,3]");

String xml3 = xmlSerializer.write(json3);

System.out.println("xml3:" + xml3);

输出：xml3:<?xml version="1.0" encoding="UTF-8"?>

<a><e type="number">1</e><e type="number">2</e><e type="number">3</e></a>

|  |
| --- |
| 环境：  　　JDK5  　　json-lib-2.3-jdk15  　　测试所依赖的包：  　　json-lib-2.3-jdk15.jar  　　commons-collections.jar  　　commons-lang.jar  　　commons-logging.jar  　　commons-beanutils.jar  　　ezmorph-1.0.6.jar  　　xom-1.1.jar  　　测试代码：  　　import net.sf.json.JSONArray;  　　import net.sf.json.JSONObject;  　　import net.sf.json.xml.XMLSerializer;  　　import java.util.ArrayList;  　　import java.util.HashMap;  　　import java.util.List;  　　import java.util.Map;  　　/\*\*  　　\* JSON for Java测试代码  　　\*  　　\*/  　　public class TestJSON {  　　public static void main(String[] args) {  　　test1();  　　test2();  　　test3();  　　test4();  　　test5();  　　}  　　/\*\*  　　\* 数组或集合-->JSON串  　　\*/  　　public static void test1() {  　　System.out.println("------------数组或集合-->JSON串----------");  　　boolean[] boolArray = new boolean[]{true, false, true};  　　JSONArray jsonArray1 = JSONArray.fromObject(boolArray);  　　System.out.println(jsonArray1);  　　//[true,false,true]  　　List list = new ArrayList();  　　list.add("first");  　　list.add("second");  　　JSONArray jsonArray2 = JSONArray.fromObject(list);  　　System.out.println(jsonArray2);  　　//["first","second"]  　　JSONArray jsonArray3 = JSONArray.fromObject("['json','is','easy']");  System.out.println(jsonArray3);  　　// ["json","is","easy"]  　　}  　　/\*\*  　　\* Object|Map-->JSON串  　　\*/  　　public static void test2() {  　　System.out.println("------------Object|Map-->JSON串----------");  　　Map map = new HashMap();  　　map.put("name", "json");  　　map.put("bool", Boolean.TRUE);  　　map.put("int", new Integer(1));  　　map.put("arr", new String[]{"a", "b"});  　　map.put("func", "function(i){ return this.arr[i]; }");  　　JSONObject jsonObject1 = JSONObject.fromObject(map);  　　System.out.println(jsonObject1);  　　//{"func":function(i){ return this.arr[i]; },"arr":["a","b"],"int":1,"bool":true,"name":"json"}  　　JSONObject jsonObject2 = JSONObject.fromObject(new MyBean());  　　System.out.println(jsonObject2);  　　//{"func1":function(i){ return this.options[i]; },"func2":function(i){ return this.options[i]; },"name":"json","options":["a","f"],"pojoId":1}  　　}  　　/\*\*  　　\* JSON串-->Object  　　\*/  　　public static void test3() {  　　System.out.println("------------JSON串-->Object----------");  　　String json1 = "{name=\"json\",bool:true,int:1,double:2.2,func:function(a){ return a; },array:[1,2]}";  　　JSONObject jsonObject1 = JSONObject.fromObject(json1);  　　Object bean1 = JSONObject.toBean(jsonObject1);  　　System.out.println(bean1);  　　//net.sf.ezmorph.bean.MorphDynaBean@10dd1f7[  　　// {double=2.2, func=function(a){ return a; }, int=1, name=json, bool=true, array=[1, 2]}  　　//]  　　String json2 = "{bool:true,integer:1,string:\"json\"}";  　　JSONObject jsonObject2 = JSONObject.fromObject(json2);  　　BeanA bean2 = (BeanA) JSONObject.toBean(jsonObject2, BeanA.class);  　　System.out.println(bean2);  　　// BeanA{bool=true, integer=1, string='json'}  　　}  　　/\*\*  　　\* JSON串-->XML  　　\*/  　　public static void test4() {  　　System.out.println("------------JSON串-->XML----------");  　　JSONObject json = new JSONObject(true);  　　String xml = new XMLSerializer().write(json);  　　System.out.println(xml);  　　JSONObject json1 = JSONObject.fromObject("{\"name\":\"json\",\"bool\":true,\"int\":1}");  　　String xml1 = new XMLSerializer().write(json1);  　　System.out.println(xml1);  　　JSONArray json2 = JSONArray.fromObject("[1,2,3]");  　　String xml2 = new XMLSerializer().write(json2);  　　System.out.println(xml2);  　　}  /\*\*  　　\* XML-->JSON串  　　\*/  　　public static void test5() {  　　System.out.println("------------XML-->JSON串----------");  　　String xml = "" +  　　"<a class=\"array\">\n" +  　　" <e type=\"function\" params=\"i,j\">\n" +  　　" return matrix[i][j];\n" +  　　" </e>\n" +  　　"</a>";  　　JSONArray json = (JSONArray) new XMLSerializer().read(xml);  　　System.out.println(json);  　　}  　　}  　　import net.sf.json.JSONFunction;  　　/\*\*  　　\* 测试Bean  　　\*  　　\*/  　　public class MyBean {  　　private String name = "json";  　　private int pojoId = 1;  　　private char[] options = new char[]{'a', 'f'};  　　private String func1 = "function(i){ return this.options[i]; }";  　　private JSONFunction func2 = new JSONFunction(new String[]{"i"}, "return this.options[i];");  　　public String getName() {  　　return name;  　　}  　　public void setName(String name) {  　　this.name = name;  　　}  　　public int getPojoId() {  　　return pojoId;  　　}  　　public void setPojoId(int pojoId) {  　　this.pojoId = pojoId;  　　}  　　public char[] getOptions() {  　　return options;  　　}  　　public void setOptions(char[] options) {  　　this.options = options;  　　}  　　public String getFunc1() {  　　return func1;  　　}  public void setFunc1(String func1) {  　　this.func1 = func1;  　　}  　　public JSONFunction getFunc2() {  　　return func2;  　　}  　　public void setFunc2(JSONFunction func2) {  　　this.func2 = func2;  　　}  　　}  　　/\*\*  　　\* 测试Bean  　　\*  　　\*/  　　public class BeanA {  　　// String json2 = "{bool:true,integer:1,string:\"json\"}";  　　private boolean bool;  　　private Integer integer;  　　private String string;  　　public boolean isBool() {  　　return bool;  　　}  　　public void setBool(boolean bool) {  　　this.bool = bool;  　　}  　　public Integer getInteger() {  　　return integer;  　　}  　　public void setInteger(Integer integer) {  　　this.integer = integer;  　　}  　　public String getString() {  　　return string;  　　}  　　public void setString(String string) {  　　this.string = string;  　　}  　　@Override  　　public String toString() {  　　return "BeanA{" +  　　"bool=" + bool +  　　", integer=" + integer +  　　", string='" + string + '\'' +  　　'}';  　　}  　　}  　　运行结果：  　　------------数组或集合-->JSON串----------  　　[true,false,true]  　　["first","second"]  　　["json","is","easy"]  　　------------Object|Map-->JSON串----------  　　{"func":function(i){ return this.arr[i]; },"arr":["a","b"],"int":1,"bool":true,"name":"json"}  　　{"func1":function(i){ return this.options[i]; },"func2":function(i){ return this.options[i]; },"name":"json","options":["a","f"],"pojoId":1}  　　------------JSON串-->Object----------  　　net.sf.ezmorph.bean.MorphDynaBean@bd0108[  　　{double=2.2, func=function(a){ return a; }, int=1, name=json, bool=true, array=[1, 2]}  　　]  　　BeanA{bool=true, integer=1, string='json'}  　　------------JSON串-->XML----------  　　<?xml version="1.0" encoding="UTF-8"?>  　　<o null="true"/>  　　<?xml version="1.0" encoding="UTF-8"?>  　　<o><bool type="boolean">true</bool><int type="number">1</int><name type="string">json</name></o>  　　<?xml version="1.0" encoding="UTF-8"?>  　　<a><e type="number">1</e><e type="number">2</e><e type="number">3</e></a>  　　------------XML-->JSON串----------  　　2009-12-28 15:20:39 net.sf.json.xml.XMLSerializer getType  　　信息: Using default type string  　　[function(i,j){ return matrix[i][j]; }]  　　Process finished with exit code 0  　　特别注意：  　　1、所有的Bean都应该定义为public，否则会出现  　　net.sf.json.JSONException: java.lang.NoSuchMethodException: Property '\*\*\*' has no getter method  　　的错误。  　　2、必须引入xom-1.1.jar包，否则抛出  　　java.lang.NoClassDefFoundError: nu/xom/Serializer  　　的异常。  　　另外，还有个开源的JSON库，推荐下：  　　http://code.google.com/p/google-gson/ |