# 解析JSON为Object算法文档

## 算法描述：

设计一个算法，将一个JSON字符串转化为一个已知结构的对象或数组。

## 算法流程

1. 如果接下来要解析的是一个对象，则递归解析该对象，解析完该对象再返回当前对象或数组。
2. 如果接下来要解析的是一个数组，则按解析数组的方法解析数组。
3. 如果接下来要解析的是一个基本数据类型且不是第一次解析，则直接解析。
4. 重复以上步骤1，2，3直到JSON字符串完全解析结束。

## 3.处理多维数组

**JSON规定：**

1. JSON数组内部之内存储基本数据类型，不能有对象。
2. JSON数组内部的成员变量的类型可以不相同。

**使用java代码实现算法时：**

**人为规定：**

1. 所有的字段都是以setXXX()方法注入，与字段间接联系。
2. 取第一个有一个参数的set方法注入值。
3. 允许不接收JSON的某个字段（不设置set方法），会打印信息不忽略字段提示。
4. JSON数组内部的所有成员的类型一致，且为java的基本数据类型。
5. JSON数组的维度不超过256。
6. 不允许有空数组。

## JOSN一些符号的作用

|  |  |  |
| --- | --- | --- |
| 序号 | 符号 | 作用 |
| 1 | { | 一个对象的开始 |
| 2 | } | 一个对象的结束 |
| 3 | “ | 一个字符串的开始或结束 |
| 4 | , | 成员分隔符 |
| 5 | [ | 一个数组的开始 |
| 6 | ] | 一个数组的结束 |
| 7 | : | 后接一个成员属性的值，标志一个键值对（ key : value ） |
| 8 | 空格，\t，\n | 在字符串外时起分隔作用，字符串内则表示一个字符 |
| 9 | - | 键值对（ key : value ）的值是负数 |
| 10 | [0 ... 9] | 键值对（ key : value ）的值是数字类型 |
| 11 | t|f | 键值对（ key : value ）的值可能是逻辑类型 |
| 12 | n | 键值对（ key : value ）的值可能是null |
| 13 | 其他字符 |  |

## JSON类型

将JSON的6种数据类型：数字，字符串，逻辑值，数组，对象，null

解析类型转换方法：

|  |  |
| --- | --- |
| JSON类型 | 自动转换java类型 |
| 数字类型 | 自动解析为BigDeciaml，再转为set方法的参数类型 |
| 字符串类型 | 字符串类型（不用转换） |
| 逻辑类型 | 布尔类型：true|false |
| null类型 | 空值类型null |
| 对象类型 | 对象类型（不用转换） |
| 数组类型 | 转多级链表，再转规范式数组。 |

表1

## 伪代码

**def** 全局属性

def beiGaoChars, current

**def** 清除分隔符方法

while current < beiGaoChars.length 且 \

beiGaoChars[current] 属于 {**‘ ’**, **‘\t’**, **‘\n’**}

++current

**def** 解析字符串方法

++current

begin ← current

while current 小于 beiGaoChars.length 且 \

beiGaoChars[current] 不是 ‘\”’

++current

if current 小于 beiGaoChars.length 且 \

beiGaoChars[current] 是 ‘\”’ 且 \

current - 1 大于 begin

end ← current - 1

返回beiGaoChars[begin ... end]

else

抛出JSON格式错误异常

**def** 解析数字类型方法

传入tClass

begin ← current

while current < beiGaoChars.length 且 \

beiGaoChars[current] 不属于 {**‘ ’**, **‘,’**, **‘]’**, **‘}’**}

++current

If current 等于 beiGaoChars.length

抛出JSON格式异常

else

end ← current - 1

num ← 转化beiGaoChars[begin ... end]为tClass类型的数字

返回 num

**def** 解析逻辑类型方法

if current + 3 小于 beiGaoChars.length 且

beiGaoChars[index ... index + 3] 是 “true”

current += 4

返回true

else if current + 4 小于 beiGaoChars.length 且

beiGaoChars[index ... index + 4] 是 “false”

current += 5

返回false

eles

抛出JSON格式错误异常

if current >= current

抛出JSON格式错误异常

**def** 解析空值方法

current += 4

def 节点类型Node

成员 o，线性表list

**def** 解析数组方法

传入tClass

将tClass转变为基本类型

array[] ← 创建记录各个维度最大下标的数组

stack ← 创建栈

list ← 创建节点类型的线性表

stack压栈0

stack压栈0

++current

oldChar = ‘[‘

while current 小于 beiGaoChars.length 且

stack.size 大于 1

调用清除分隔符方法

if beiGaoChars[current] 是 ‘[‘

stack压栈0

olcChar ← beiGaoChars[current++]

if beiGaoChars[current] 是 ‘]’

if oldChar 是 ‘[‘

抛出空数组异常

stack出栈

stack栈顶元素+1

array[stack.size - 1] = max（array[stack.size - 1], stack栈顶元素）

if beiGaoChars[current] 是 ‘,’

olcChar = beiGaoChars[current++]

else

olcChar = beiGaoChars[current++]

o = 调用解析键值方法

node ← 创建节点

list1 ← 创建线性表

list1[0 ... stack.size - 3]← stack[1 ... stack.size - 2]

list1[stack.size - 2] = stack[stack.size - 1] - 1

设置 o，list1进node

添加node到list

if stack.size 不等于 1

抛出JSON格式错误异常

end ← 除去array中后面连续0的位数，然后取剩余位数

obj ← 创建多维数组，以array[1 ... end - 1] 为维数。

for node in list

list1 ← 获取node中的list

obj[ list1[0] ] [ list1[1] ] ... [ list1[list1.length - 1] ] ← 获取node中的o

返回 obj

**def** 解析对象方法

传入tClass

obj ← 创建tClass类型的对象

do

调用清除分隔符方法

if beiGaoChars[current] 不属于 {**‘{’**，**‘,’**}

break

++current

keyName← 调用解析键名方法

调用清除分隔符方法

if current 大于等于 beiGaoChars.length 或 \

beiGaoChars[current] != ‘:’

抛出JSON格式错误异常

o ← 调用解析键值方法

添加键值对（keyName：o）到对象obj。

while true

调用清除分隔符方法

if current 小于 beGaoChars[current] 且 \

beGaoChars[current] 是 ‘}’

++current

返回 obj

else

抛出JSON格式错误异常

**def** 解析键名方法

s ← 调用解析字符串方法

返回s

**def** 解析键值方法

调用清除分隔符方法

if current 大于等于 beiGaoChars.length

抛出JSON格式错误异常

switch beiGaoChars[current]

case ‘{’

obj ← 调用解析对象方法

case ‘\”’

obj ← 调用解析字符串方法

case ‘[’

obj ← 调用解析数组方法

case [‘-’, ‘0’ ... ‘9’]

obj ← 调用解析数字方法

case [‘f’, ‘t’]

if beGaoChars[current ... current + 3] 是 true 或

beGaoChars[current ... current + 4] 是 false

obj ← 调用解析逻辑类型方法

else

抛出JSON格式异常

case ‘n’

if beiGaoChars[current ... current + 3]是null

obj ← 调用解析空值方法

else

抛出JSON格式异常

default

抛出JSON格式异常

返回obj

**def** JSON转Pojo入口方法

传入一个json，tClass

beiGaoChars[] ← 转化Json为字符数组

current ← 0

obj ← 调用解析对象方法

返回 obj

**def** JSON转Array入口方法

传入一个json，tClass

beiGaoChars[] ← 转化json为字符数组

current ← 0

array[]← 调用解析数组方法

返回 array[]

## 软件测试

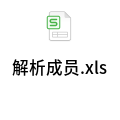
### 测试解析单个成员

测试方法：黑盒测试（等价类划分、边界值分析）

测试对象：JSON内对个键值对（key：value）

输入数据格式：JSON数据

输出数据格式：User类



### 测试解析二层嵌套JSON

测试对象：含二层嵌套的JSON

输入数据格式：JSON数据

输出数据格式：User类

