一个基于 JSON 的对象序列化算法

张 涛[1,2],黄 强[3],毛磊雅[1],高 兴[1]
ZHANG Tao[1,2],HUANG Qiang[3],MAO Lei-ya[1],GAO Xing[1]

1.西南交通大学 信息科学与技术学院,成都 610031
2.四川师范大学 计算机软件重点实验室,成都 610068
3.四川农业大学 信息工程与技术学院,四川 雅安 625014
1.Department of Information Science and Technology,Southwest Jiaotong University,Chengdu 610031,China
2.Computer Software Lab,Sichuan Normal University,Chengdu 610068,China
3.Information & Engineering Technology College,Sichuan Agricultural University,Ya'an,Sichuan 625014,China
E-mail:tzhang2000@126.com

ZHANG Tao,HUANG Qiang,MAO Lei-ya,et al.Algorithm of object serialization based on JSON.Computer Engineering and Applications,2007,43(15):98-100.

Abstract: Nowadays,XML is mainly applied to data-interchange in current Web application system which is based on Ajax (Asynchronous JavaScript and XML) technology,but as a structured document,XML must be manually resolved between the server-side and the client-side,so that more system resources is consumed,and therefore problems such as weak performance, deficiency of compatibility and lack of sensitivity are caused by using XML. JSON (JavaScript Object Notation) is a lightweight data-interchange format,and it can be easily parsed by browser which supports JavaScript.In this paper,an algorithm of object serialization based on JSON is proposed,by which the JSON grammar is analyzed and the object navigation graph is also established.With this algorithm,the Java objects can be translated into JSON expressions transparently,so as to the client-side can take full advantage of JavaScript engine to parse JSON expressions,thus the problems resulted from XML can be solved effectively.

Key words: Ajax;data-interchange;Java Script Object Notation(JSON);serialization;object navigation graph

摘 要:目前基于 Ajax 技术的 Web 开发主要采用 XML 进行数据交换,然而 XML 是一种结构化的文档,需要服务器和客户端都对其进行手工解析,将会占用更多的系统资源,因此采用 XML 进行数据交换会导致性能低下、兼容性不够、灵敏度低的问题。JSON (JavaScript Object Notation)是一种轻量级的数据交换格式,易于被支持 JavaScript 的浏览器所解析。提出了一种基于 JSON 的对象序列化算法,该算法通过分析 JSON 文法并建立对象导航图,透明地将 Java 对象序列化成 JSON 表达式,使客户端能够很好地利用 JavaScript 引擎来解析 JSON 响应,有效地解决了解析 XML 所造成的缺陷。

关键词:Ajax;数据交换;JSON;序列化;对象导航图
文章编号:1002-8331(2007)15-0098-03 **文献标识码**:A **中图分类号**:TP311.52

1 引言

作为 Web 2.0 规范的核心组成部分,Ajax 技术通过使用浏览器端的 JavaScript、DHTML 和与服务器异步通信机制很好地弥补了 B/S 系统在 Web 界面上对用户自然和响应灵敏度方面的不足(使用户从请求/响应的循环中解脱出来。借助于 Ajax,可以在用户单击按钮时,使用 JavaScript 和 DHTML 立即更新 UI,并向服务器发出异步请求,服务器根据请求完成业务逻辑。当请求返回时,Ajax 会使用 JavaScript 和 CSS 动态更新 UI,而不是刷新整个视图。在整个过程中,用户甚至不知道浏览器正在与服务器通信,让 B/S 系统实现和 C/S 系统一样的即时响应)。然而 Ajax 的数据交换主要基于 XML,由于 XML 是一种结构文档, 需要服务器和客户端都对其进行手工生成和解析,为系统开发带来了诸多不便, 并且 JavaScript 基于 DOM 的 XML 解析模型使其效率低下并且需要占用较多的系统资源,所以不能很好地融入基于 OO 的服务系统。JSON 是一种轻量级的数据交换格式,易于被支持 JavaScript 的浏览器所解析,并且易于维护。本文提出了一种基于 JSON(JavaScript Object Notation)的对象序列化算法,该算法通过对象图导航和 JSON 文法,将服务系统所产生的处理结果对象图(Object Graph)从服务器序列化到客户端并被浏览器反序列化到其运行环境中,

基金项目:四川师范大学 Sphinx 交互式化学 CAI 系统(省部级科学基金)(No.00060510)。
作者简介:张涛(1981-),男,回族,研究生,主要研究方向:网络与信息系统,软件工程;黄强(1981-),男,讲师,主要研究方向:软件工程,虚拟机,人工智能,神经网络;毛磊雅(1982-),女,研究生,主要研究方向:计算机网络及 SUPANET 体系结构;高兴(1981-),女,研究生,主要研究方向:计算机网络与信息系统。

使客户端能够使用 JavaScript 透明地处理服务器端产生的结果对象，并使用 JSON 作为桥梁完全消除了 XML 的产生和解析过程。

2　基于 JSON 的对象序列化算法

2.1　JSON 文法简介

JSON 是一种轻量级的数据交换格式。易于人阅读和编写，同时也易于机器解析和生成。它基于 JavaScript Programming Language,Standard ECMA-262 3rd Edition-December 1999 的一个子集。JSON 采用完全独立于语言的文本格式,但是也使用了类似于 C 语言家族的习惯。由于 JSON 是 JavaScript 规范的子集,所以 JSON 文档能被支持 JavaScript 的浏览器所解析,这些特性使 JSON 成为理想的数据交换语言。JSON 所定义的文法如下所示:

object
　　{},{members}
members
　　string:value;
　　members,string:value
array
　　[]
　　[elements]
elements
　　value
　　elements,value
value
　　string,number,object,
　　array,true,false,null

JSON 建构于两种结构:

(1)“名称/值”对的集合(A collection of name/value pairs)。不同的语言中,它被理解为对象(object),纪录(record),结构(struct),字典(dictionary),哈希表(hash table),有键列表(keyed list),或者关联数组(associative array)。

(2)值的有序列表(An ordered list of values)。在大部分语言中,它被理解为数组(array)。

这些都是常见的数据结构。JSON 具有以下这些形式:

对象是一个无序的“‘名称/值’对”集合。一个对象以“{”(左括号)开始,“}”(右括号)结束。每个“名称”后跟一个“:”(冒号);“‘名称/值’对”之间使用“,”(逗号)分隔。

数组是值(value)的有序集合。一个数组以“[”(左中括号)开始,“]”(右中括号)结束。值之间使用“,”(逗号)分隔。

值(value)可以是双引号括起来的字符串(string)、数值(number)、true、false、null、对象(object)或者数组(array)。这些结构可以嵌套。

字符串(string)是由双引号包围的任意数量 Unicode 字符的集合,使用反斜线转义。一个字符(character)即一个单独的字符串(character string)。

2.2　基于 JSON 的对象序列化算法的设计

Java 对象和 JavaScript 对象之间的交换结构如图 1 所示。

建立 Java 和 JSON 类型的映射关系如表 1 所示。

其中 Scalar 类型为 JSON 的基础类型，与 Scalar 对应的 Java 对象可以被直接转换为等值的 JSON 描述字符串，例如 Java 的整形 5 被转换为 JSON 表达式“5”,而 Java 的集合

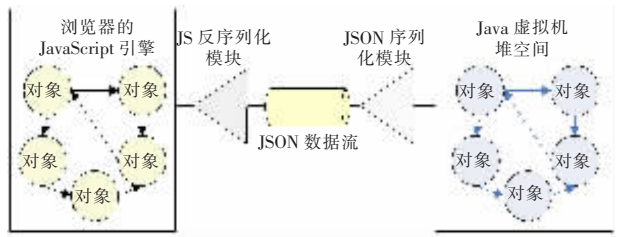

图 1　Java 对象和 JavaScript 对象之间的交换结构

表 1　数据类型映射表

Java 类型	JSON 类型
byte,short,int,long,float,double,Boolean,String,Date	Scalar
Collection,Array	Array
Map	Object
Bean	Object

(Collection)和数组(Array)可以转换为等价的 JSON 数组对象,而 Java 的映射(Map)在 JSON 中会被转换为对象,Map 的键作为对象的属性(Attribute),而 Map 的值作为该属性的值;而 JavaBean 也会被转换成 JSON 的对象，和 Map 不同,JavaBean 和 JSON 对象的逻辑结构一致,Bean 属性和 JSON 对象的属性一一对应。

本文的序列化算法将 JSON 的表达式拆分成两种 JavaScript 原语:

JS 变量定义(JS Var Definition):var $a=b$;

JS 变量赋值(JS Var Assignment):$a=b$;

该算法中,中间变量 b 为前缀加随机数格式来确保对象唯一性，例如:_sj_24825485，特别的，根对象的表示为 _sj_root_2089889313。

该算法将 Java 对象与 JSON 对象的转换定义为变量定义(JSD)和变量赋值(JSA)两部分,在对对象图的一个节点进行解析前会输出一条 JSD,在解析完成以后输出一条 JSA;针对该节点的对象属性类型,当处理标量类型时,直接使用标量转换,标量转换的输出为 Scalar 字符串(JSS),当处理复合对象时，采用引用占位符来替换标量转换，输出为引用占位符(JSR),并且重新进入节点解析过程(根据节点类型采用相应的序列化方法),整个对象的序列化流程如图 2 所示。

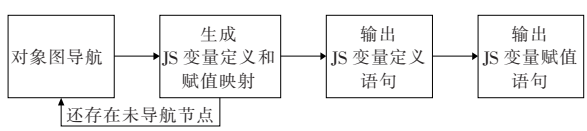

图 2　对象序列化流程图

由于 Java 的对象可能是复合对象引用，并且引用之间可以存在任意层次的嵌套，所以使得对象图成为一个有向有环图,故要动态的对象导航图(Navigation Graph),须建立基于类型的转换状态机 $M(\text{json})=(Q,S,T,f,g,N)$ 如图 3 所示。

其中 Q 是有限状态集;T 是触发事件集合;S 是输入序列集合，作为输入集;f 是状态转换函数 $f:Q*S->Q$;g 为输出集;$N\in Q$ 是初态。

定义　节点解析前均为 N 态

状态集 Q={

N　　中间节点

A　　数组型的节点

C　　集合型的节点

M　　映射型的节点

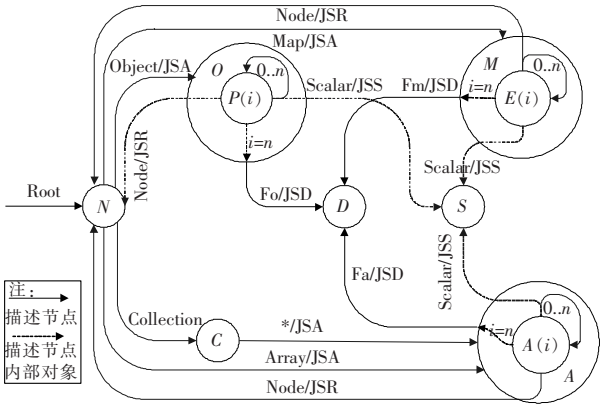

图 3 转换状态机

O JavaBean 型的节点

D 变量定义集合

S 标量输出集合

$A(i)$ 数组的元素,$i\in\{0,1,2,\cdots,n\}$

$E(i)$ 映射的入口,$i\in\{0,1,2,\cdots,n\}$

$P(i)$ JavaBean 型的属性,$i\in\{0,1,2,\cdots,n\}$

}

触发事件集={

Fo JavaBean 型节点解析完成事件

Fm 映射型节点解析完成事件

Fa 数组型节点解析完成事件

}

输入集={

Array 输入数组型节点

Collection 输入集合型节点

Map 输入映射型节点

Object 输入 JavaBean 型节点

Scalar 节点中内部对象的属性为 Scalar 类型

Node 节点中内部对象的属性为非 Scalar 类型(复合类型)

}

输出集={

JSD 变量定义

JSR 引用占位符

JSA 变量赋值

JSS 标量输出

}

状态转换函数:

$f(N,\text{Object})=O$,$f(N,\text{Array})=A$,$f(N,\text{Collection})=C$,$f(N,\text{Map})=M$,$f(C,*)=A$,$f(A(i),\text{Scalar})=S$,

$f(E(i),\text{Scalar})=S$,$f(P(i),\text{Scalar})=S$,$f(A(i),\text{Node})=N$,$f(E(i),\text{Node})=N$,$f(P(i),\text{Node})=N$,

$f(A,Fa)=D$,$f(M,Fm)=D$,$f(O,Fo)=D$

输出函数:

$g(A,\text{Scalar})=\text{JSS}$,$g(M,\text{Scalar})=\text{JSS}$,$g(O,\text{Scalar})=\text{JSS}$,$g(A,\text{Node})=\text{JSR}$,$g(M,\text{Node})=\text{JSR}$,$g(O,\text{Node})=\text{JSR}$

$g(A,\text{Fa})=\text{JSD}$,$g(M,\text{Fm})=\text{JSD}$,$g(O,\text{Fo})=\text{JSD}$,$g(N,\text{Object})=\text{JSA}$,$g(N,\text{Array})=\text{JSA}$,$g(N,\text{Map})=\text{JSA}$

注:

0..n 表示对节点的内部对象进行循环解析

i=n 表示某节点的所有内部对象都已经解析完成

该状态机模型实现的算法能够导航任意复杂的 Java 对象导航图,并生成符合 JSON 文法的 Java 对象图所对等的 JSON 表达式集合。

2.3 基于 JSON 的对象序列化算法的应用实例

以下面图 4 中的 Java 对象为例,该 Java 对象为一个 JavaBean 类型的对象,首先进入状态机的 N 状态,通过状态机的状态转换函数 $f(N,object)=O$,跳转到 O 状态,并输出一条 JSA,在该状态下进行内部节点的解析;其次,由于该对象的内部节点均属于 JSON 文法中的 Scalar 类型,故通过 $f(P(i),Scalar)=S$,进行标量输出;最后当内部节点解析完成时(i=n),激活触发事件 Fo,通过 $f(O,Fo)=D$,输出一条 JSD,并输出一条 JSA。至此该 JavaBean 型节点解析完成。得到的 JSON 序列如图 5 所示。

```
public static Item getItem() {
    Item iteml=new Item();
    iteml.setId("001");
    iteml.setName("CF Card");
    iteml.setPrice("$49");
    return iteml;
}
```

图 4 Java 对象示例图

```
var_sj_root_2089889313;
var_sj_12848256=[null];
var_sj_26130918={"  +id:"001",
      "name:"CF Card",price:"$49"};
_sj_12848256[0]=_sj_26130918;
_sj_root_2089889313=_sj_12848256;
_sj_root_2089889313;
```

图 5 JSON 响应序列

客户端通过 JavaScript 引擎中的 eval()函数反序列化图 5 中的 JSON 响应序列,使之运行在客户的浏览器上。

3 JSON 与 XML 的比较

JSON 比较简单,易于读写,格式都是压缩的,占用带宽小。采用 JSON 作为数据交换格式,客户端只需利用 JavaScript 中的 eval()函数进行解析(而且仅仅需要一行代码),显然要比基于 DOM 的 XML 解析模型更简单,而且速度更快。表 2 为 JSON 与 XML 之间的比较。

表 2 JSON 与 XML 之间的比较

	数据格式	占用带宽	代码开发量	兼容性	解析速度	开发效率	灵敏度
JSON	简单	小	小	较强	快	高	高
XML	复杂	大	大	一般	慢	低	低

4 总结

本文提出了一种基于 JSON 的对象序列化算法,该算法通过分析 JSON 文法并建立对象导航图,透明地将 Java 对象序列化成 JSON 对象,使客户端利用 JavaScript 引擎来解析 JSON 响应,有效解决了解析 XML 所造成的缺陷。

(收稿日期:2006 年 10 月)

参考文献:

[1] Garrett J J.ajax:a new approach to web applications [EB/OL].(2005). http://www.adaptivepath.com/publications/essays/archives/000385.php.

[2] Smith K.Simplifying Ajax-style Web development[J].Computer, 2006,39(5):98-101.

[3] Paulson L D.Building rich web applications with Ajax[J].Computer,2005,38(10):14-17.

[4] Sayar A,Pierce M,Fox G.Integrating AJAX approach into GIS visualization web services[C]//Telecommunications,AICT-ICIW'06: International Conference on Internet and Web Applications and

(下转 133 页)