[**JSON入门教程**](http://blog.csdn.net/terryzero/article/details/4015597)

2009-03-22 23:17 13876人阅读 [评论](http://blog.csdn.net/terryzero/article/details/4015597#comments)(19) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/terryzero/article/details/4015597#report)

[json](http://www.csdn.net/tag/json)[javascript](http://www.csdn.net/tag/javascript)[object](http://www.csdn.net/tag/object)[string](http://www.csdn.net/tag/string)[服务器](http://www.csdn.net/tag/%e6%9c%8d%e5%8a%a1%e5%99%a8)[xml](http://www.csdn.net/tag/xml)

尽管有许多宣传关于 XML 如何拥有跨平台，跨语言的优势，然而，除非应用于 Web Services，否则，在普通的 Web 应用中，开发者经常为 XML 的解析伤透了脑筋，无论是服务器端生成或处理 XML，还是客户端用 JavaScript 解析 XML，都常常导致复杂的代码，极低的开发效率。实际上，对于大多数 Web 应用来说，他们根本不需要复杂的 XML 来传输数据，XML 的扩展性很少具有优势，许多 AJAX 应用甚至直接返回 HTML 片段来构建动态 Web 页面。和返回 XML 并解析它相比，返回 HTML 片段大大降低了系统的复杂性，但同时缺少了一定的灵活性。

　　现在， JSON 为 Web 应用开发者提供了另一种数据交换格式。让我们来看看 JSON 到底是什么，同 XML 或 HTML 片段相比，JSON 提供了更好的简单性和灵活性。

　　JSON 数据格式解析

　　和 XML 一样，JSON 也是基于纯文本的数据格式。由于 JSON 天生是为 JavaScript 准备的，因此，JSON 的数据格式非常简单，您可以用 JSON 传输一个简单的 String，Number，Boolean，也可以传输一个数组，或者一个复杂的 Object 对象。

　　String，Number 和 Boolean 用 JSON 表示非常简单。例如，用 JSON 表示一个简单的 String “ abc ”，其格式为：

"abc"

　　除了字符 "，，/ 和一些控制符（b，f，n，r，t）需要编码外，其他 Unicode 字符可以直接输出。

一个 Number 可以根据整型或浮点数表示如下：

　这与绝大多数编程语言的表示方法一致，例如：

12345（整数）  
-3.9e10（浮点数）

　　Boolean 类型表示为 true 或 false 。此外，JavaScript 中的 null 被表示为 null，注意，true、false 和 null 都没有双引号，否则将被视为一个 String 。

　　JSON 还可以表示一个数组对象，使用 [] 包含所有元素，每个元素用逗号分隔，元素可以是任意的 Value，例如，以下数组包含了一个 String，Number，Boolean 和一个 null：

["abc",12345,false,null]

　　Object 对象在 JSON 中是用 {} 包含一系列无序的 Key-Value 键值对表示的，实际上此处的 Object 相当于 Java 中的 Map<String, Object>，而不是 Java 的 Class 。注意 Key 只能用 String 表示。

　　例如，一个 Address 对象包含如下 Key-Value：

city:Beijing  
street:Chaoyang Road  
postcode:100025（整数）

　　用 JSON 表示如下：

{"city":"Beijing","street":" Chaoyang Road ","postcode":100025}

　　其中 Value 也可以是另一个 Object 或者数组，因此，复杂的 Object 可以嵌套表示，例如，一个 Person 对象包含 name 和 address 对象，可以表示如下：

{"name":"Michael","address":  
　　{"city":"Beijing","street":" Chaoyang Road ","postcode":100025}  
}

JavaScript 处理 JSON 数据

　　上面介绍了如何用 JSON 表示数据，接下来，我们还要解决如何在服务器端生成 JSON 格式的数据以便发送到客户端，以及客户端如何使用 JavaScript 处理 JSON 格式的数据。

　　我们先讨论如何在 Web 页面中用 JavaScript 处理 JSON 数据。我们通过一个简单的 JavaScript 方法就能看到客户端如何将 JSON 数据表示给用户：

function handleJson() {  
　var j={"name":"Michael","address":  
　　　{"city":"Beijing","street":" Chaoyang Road ","postcode":100025}  
　};  
　document.write(j.name);  
　document.write(j.address.city);  
}

　　假定服务器返回的 JSON 数据是上文的：

{"name":"Michael","address":  
　　{"city":"Beijing","street":" Chaoyang Road ","postcode":100025}  
}

　　只需将其赋值给一个 JavaScript 变量，就可以立刻使用该变量并更新页面中的信息了，相比 XML 需要从 DOM 中读取各种节点而言，JSON 的使用非常容易。我们需要做的仅仅是发送一个 Ajax 请求，然后将服务器返回的 JSON 数据赋值给一个变量即可。有许多 Ajax 框架早已包含了处理 JSON 数据的能力，例如 Prototype（一个流行的 JavaScript 库：http://prototypejs.org）提供了 evalJSON() 方法，能直接将服务器返回的 JSON 文本变成一个 JavaScript 变量：

new Ajax.Request("http://url", {  
　method: "get",  
　onSuccess: function(transport) {  
　　var json = transport.responseText.evalJSON();  
　　// TODO: document.write(json.xxx);  
　}  
});

服务器端输出 JSON 格式数据

　　下面我们讨论如何在服务器端输出 JSON 格式的数据。以 Java 为例，我们将演示将一个 Java 对象编码为 JSON 格式的文本。

　　将 String 对象编码为 JSON 格式时，只需处理好特殊字符即可。另外，必须用 (") 而非 (') 表示字符串：

**[java]** [view plaincopyprint?](http://blog.csdn.net/terryzero/article/details/4015597)

1. static String string2Json(String s) {
2. StringBuilder sb = new StringBuilder(s.length()+20);
3. sb.append('"');
4. for (int i=0; i<s.length(); i++) {
5. char c = s.charAt(i);
6. switch (c) {
7. case '"':
8. sb.append(""");
9. break;
10. case '':
11. sb.append("");
12. break;
13. case '/':
14. sb.append("/");
15. break;
16. case 'b':
17. sb.append("");
18. break;
19. case 'f':
20. sb.append("f");
21. break;
22. case 'n':
23. sb.append("
24. ");
25. break;
26. case 'r':
27. sb.append("
28. ");
29. break;
30. case 't':
31. sb.append("　　");
32. break;
33. default:
34. sb.append(c);
35. }
36. }
37. sb.append('"');
38. return sb.toString();
39. }



将 Number 表示为 JSON 就容易得多，利用 Java 的多态，我们可以处理 Integer，Long，Float 等多种 Number 格式：

static String number2Json(Number number) {  
　　return number.toString();  
}

　　Boolean 类型也可以直接通过 toString() 方法得到 JSON 的表示：

static String boolean2Json(Boolean bool) {  
　　return bool.toString();  
}

　　要将数组编码为 JSON 格式，可以通过循环将每一个元素编码出来：

**[java]** [view plaincopyprint?](http://blog.csdn.net/terryzero/article/details/4015597)

1. static String array2Json(Object[] array) {
2. if (array.length==0)
3. return "[]";
4. StringBuilder sb = new StringBuilder(array.length << 4);
5. sb.append('[');
6. for (Object o : array) {
7. sb.append(toJson(o));
8. sb.append(',');
9. }
10. // 将最后添加的 ',' 变为 ']':
11. sb.setCharAt(sb.length()-1, ']');
12. return sb.toString();
13. }



　　最后，我们需要将 Map<String, Object> 编码为 JSON 格式，因为 JavaScript 的 Object 实际上对应的是 Java 的 Map<String, Object> 。该方法如下：

**[java]** [view plaincopyprint?](http://blog.csdn.net/terryzero/article/details/4015597)

1. static String map2Json(Map<String, Object> map) {
2. if (map.isEmpty())
3. return "{}";
4. StringBuilder sb = new StringBuilder(map.size() << 4);
5. sb.append('{');
6. Set<String> keys = map.keySet();
7. for (String key : keys) {
8. Object value = map.get(key);
9. sb.append('"');
10. sb.append(key);
11. sb.append('"');
12. sb.append(':');
13. sb.append(toJson(value));
14. sb.append(',');
15. }
16. // 将最后的 ',' 变为 '}':
17. sb.setCharAt(sb.length()-1, '}');
18. return sb.toString();
19. }



为了统一处理任意的 Java 对象，我们编写一个入口方法 toJson(Object)，能够将任意的 Java 对象编码为 JSON 格式：

**[java]** [view plaincopyprint?](http://blog.csdn.net/terryzero/article/details/4015597)

1. public static String toJson(Object o) {
2. if (o==null)
3. return "null";
4. if (o instanceof String)
5. return string2Json((String)o);
6. if (o instanceof Boolean)
7. return boolean2Json((Boolean)o);
8. if (o instanceof Number)
9. return number2Json((Number)o);
10. if (o instanceof Map)
11. return map2Json((Map<String, Object>)o);
12. if (o instanceof Object[])
13. return array2Json((Object[])o);
14. throw new RuntimeException("Unsupported type: " + o.getClass().getName());
15. }



　　我们并未对 Java 对象作严格的检查。不被支持的对象（例如 List）将直接抛出 RuntimeException 。此外，为了保证输出的 JSON 是有效的，Map<String, Object> 对象的 Key 也不能包含特殊字符。细心的读者可能还会发现循环引用的对象会引发无限递归，例如，精心构造一个循环引用的 Map，就可以检测到 StackOverflowException：

@Test(expected=StackOverflowError.class)  
public void testRecurrsiveMap2Json() {  
　　Map<String, Object> map = new HashMap<String, Object>();  
　　map.put("key", map);  
　　JsonUtil.map2Json(map);  
}

　　好在服务器处理的 JSON 数据最终都应该转化为简单的 JavaScript 对象，因此，递归引用的可能性很小。

　　最后，通过 Servlet 或 MVC 框架输出 JSON 时，需要设置正确的 MIME 类型（application/json）和字符编码。假定服务器使用 UTF-8 编码，则可以使用以下代码输出编码后的 JSON 文本：

response.setContentType("application/json;charset=UTF-8");  
response.setCharacterEncoding("UTF-8");  
PrintWriter pw = response.getWriter();  
pw.write(JsonUtil.toJson(obj));  
pw.flush();

　　小结

　　JSON 已经是 JavaScript 标准的一部分。目前，主流的浏览器对 JSON 支持都非常完善。应用 JSON，我们可以从 XML 的解析中摆脱出来，对那些应用 Ajax 的 Web 2.0 网站来说，JSON 确实是目前最灵活的轻量级方案。