



metaparadigm

利用 JSON-RPC-Java 构建下一代 Web 应用程序

Michael Clark <michael@metaparadigm.com>
<http://oss.metaparadigm.com/jsonrpc/>

概要

● Web 交互的现状

● 更加动态的方式

- “*Weblication*”(译注：Weblication 是复合字，由Web 及Application 两字组凑而成。)概念以及“*web remoting*”

- XMLHttpRequest 对象

- 数据[反]序列化问题

● JSON、JSON-RPC 以及 JSON-RPC-Java

- 这些技术的背景

- 从 Javascript 透明的调用远程 Java 方法

● 利用 JSON-RPC-Java 构建高度动态的 web 应用程序

- 分解一个简单的范例

- 进阶主题：异步操作、类型映射

- 演示



metaparadigm

Web 交互的现状

● Web 正在延伸到更广的领域

- 在客户授权方面，创造了“inside-out”业务模型
- 无需安装特定的客户端软件，而只要一个与标准兼容的 web 浏览器。

● 落后的交互手段

- 桌面应用程序更具响应性
- 在 web 中，当按钮被点击后，我们不得不等待浏览器返回 100K 的 HTML 然后才能重新加载应用程序。

● 改进迹象



- Google Mail (Gmail)、Google Suggests





metaparadigm

“Weblication”


具有高度交互性的 Web 应用程序

-  可以和传统的桌面应用程序媲美
-  为高度易用性而设计，用户很乐意使用这些应用程序，并且长期延续。

JavaScript 和 DHTML 的深入应用

-  模仿传统用户界面而不是基于页面
-  在浏览器中运行大量的前台应用程序逻辑（而不是 JSP、ASP，等等）。

利用远程调用来避免页面重新加载

-  多数都是依赖浏览器规范来实现（比如使用 ActiveX）

“Gmail” 是个好例子



更加动态的方式

● 这些技术都是用来避免页面重新载入

- 我们可以把它叫做“web remoting”

● MSRS – 微软远程脚本

- 利用 Java applet 来提供到远程服务器的通讯
- 只能运行在支持 JVM 和 Liveconnect 的浏览器上
 - 这样就限制了平台和浏览器。

● JSRS – JavaScript 远程脚本

- 利用隐藏的框架和表单提交并从服务器获取数据
- 只支持异步操作

● .NET (SOAP , 等等)

- 只有微软的 ActiveX 才支持（如果你想生活在只有 IE 的世界里）



XMLHttpRequest 对象

- 允许你用 JavaScript 代码中发送 HTTP 请求
- 是 Web 应用程序 1.0 工作草案规范的一部分
 - Apple、Mozilla 基金会、Opera (共同发起)
 - <http://www.whatwg.org/specs/web-apps/current-work/>
- 支持众多的浏览器
 - Internet Explorer 5.0+、Mozilla 1.3+、Firefox 1.0、Safari 1.2、Opera 7.6+、Konqueror 3.3 (打补丁)、其他 Gecko browsers (Netscape、Camino、K-Meleon、Galeon)
 - 在 Macintosh 上的 IE 5.x 不支持
 - 不必担心，还有大量的浏览器可以从下面的链接中选择
 - <http://oss.metaparadigm.com/jsonrpc/browser.jsp>



XMLHttpRequest 对象 (续)

使用 XMLHttpRequest 对象

```
var http = new XMLHttpRequest();
http.open("POST", "/some-url/");
http

function getHttp() {
    // Mozilla XMLHttpRequest
    try {
        return new XMLHttpRequest();
    } catch(e) {}

    // Microsoft MSXML ActiveX
    for (var i=0; i < msxmlNames.length; i++)
    try {
        return new ActiveXObject(msxmlNames[i]);
    } catch (e) {}
}
```

微软的要复杂一些



metaparadigm

数据[反]序列化问题

- 程序员可以容易的发送和接收数据，但是我们要一种通用的格式来编码复杂的数据结构。
 - 比如这样的对等方案：从简单的以逗号分割的列到更加复杂的序列化和反序列化 schemes。
 - 应该避免每次在发送消息时书写数据格式化和解析代码，编程的最大挑战是如何更好的采用这种技术（译注：而不是花费在上面说到的内容上）。
 - 我们可以使用 XML-RPC 么？
 - XML对于这种工作还显得重了。而且还需要改进 JavaScript 的 XML 解析器才能适合跨浏览器工作。
 - 即使有了 XML-RPC 实现，仍然要使用繁琐的 API 来调用远程方法。



JSON - JavaScript 对象标记

JavaScript 事实上已经是 web 客户端的标准脚本语言

- 既然浏览器已经有这种语言了，何不从 JavaScript 应用程序发送和接收数据呢。

JavaScript 对象标记

- 可以和 C、C++、C#、Java、JavaScript、Perl、TCL 等语言绑定的轻量级数据交换格式。
- 是 ECMA-262 ECMAScript 第三版（JavaScript 1.5 或 MS Jscript 5.0）的子集
- 无需在 JavaScript 中进行反序列化解析
 - 只需分配变量
- <http://json.org/>



JSON - JavaScript 对象标记 (续)

创建以下原始类型：

Object、Array、String、Number、Boolean、null

- 对象是一组无序的参数名/参数值对的集合。对象由{ (左大括号) 开始并以} (右大括号) 结束。每个参数名都有: (冒号) 跟随，参数名/参数值对由, (逗号) 分割。
- 数组是值的有序集合。一个数组由[(左中括号) 开始并以] (右中括号) 结束。值由, (逗号) 分割。
- 值就是字符串，可以由双精度浮点数、整数、true、false、null、或者是另外一个对象或数组引用。这些结构都是可以被嵌套的。
- 字符串就是由零个和多个 Unicode 字符组成的集合，可以包装成双精度浮点数引用，由反斜杠转义。而字符也可以看作是单独的字符串。



JSON - JavaScript 对象标记 (续)

简单的 JSON 描述

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

JSON 例子 - 嵌套着数组的对象数组

```
[  
  { "country": "New Zealand",  
    "population": 3993817,  
    "animals": ["sheep", "kiwi"]  
  },  
  { "country": "Singapore",  
    "population": 4353893,  
    "animals": ["merlion", "tiger"]  
  }  
]
```



JSON-RPC - 远端过程调用

- 远端过程调用协议有点像 XML-RPC，仅仅是用 JSON 代替了 XML 作为数据序列化格式

- <http://xmlrpc.com>

- <http://json-rpc.org/>

- JSON-RPC 与 XML-RPC 比较**

- JSON 比 XML 更轻巧
 - JSON 是 JavaScript 的原生格式
 - 无需浏览器支持 JavaScript XML 解析器
 - 只需要简单的对象赋值，JavaScript 天生支持 JSON 解析

- SOAP**

- 在这里就不用多说了:)



metaparadigm

JSON-RPC - 远端过程调用（续）

```
POST /RPC2 HTTP/1.0
User-Agent: Frontier/5.1.2 (WinNT)
Host: betty.userland.com
Content-Type: text/xml
Content-length: 185

<?xml version="1.0"?>
<methodCall>
  <methodName>examples.getStateName</methodName>
  <params>
    <param>
      <value>
        <i4>41</i4>
      </value>
    </param>
  </params>
</methodCall>
```

XML-RPC 消息例子

同样的 JSON-RPC 消息

```
POST /JSON-RPC HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
Host: oss.metaparadigm.com
Content-Type: text/plain
Content-length: 59

{
  "method": "examples.getStateName",
  "params": [ 41 ]
}
```



JSON-RPC-Java

JSON-RPC-Java 是 JSON-RPC 的 Java 实现

- 可以在 Tomcat、JBoss 以及其他的 web 容器中运行

利用了 Java 反射

- 目的在于从 JavaScript 访问远程 Java 对象变得完全透明
- 输出一个对象的方法只有一行代码
- 为输出的 Java 对象在 JavaScript 中创建动态代理
- 在 Java 和 JavaScript 对象间动态映射

强大的 J2EE 安全模型

- 根据 Session 规范输出对象，可以和 JAAS 集成

广泛的浏览器支持 + Unicode

- Internet Explorer、Mozilla、Firefox、Safari、Opera 以及 Konqueror



metaparadigm

JSON-RPC-Java (续)

● 透明的序列化和反序列化 :

- 原始类型 (int、long、short、byte、boolean、char、float、double)
- 数字 (浮点、整型...) , 字符串、字符、字节数组
- Java Beans (任何含有 set 和 get 方法的对象)
- 原始类型、字符串、数字、集合类型和 Java Beans 的数组
- 异常 (尽管类型信息没有被正确保留)
- 具体的和抽象的集合类型 :
 - List、ArrayList、LinkedList 以及 Vector
 - Map、HashMap、TreeMap 以及 LinkedHashMap
 - Set、HashSet、TreeSet 以及 LinkedHashSet
 - Dictionary、Hashtable
- 以及上面列出类型的任意嵌套



JSON-RPC-Java (续)

对象代理功能

引用

- 被注册为引用的对象的类将返回一个不透明的（译注：隐含的）对象引用给 JavaScript（不是传值而是传引用）。
- 当这些不透明的参考传递给 successive Java 方法调用，将会重新组装成原来的 Java 对象（这对安全敏感的对象有利）。

可调用的引用

- 被注册为可调用引用的对象的类将返回动态代理，并允许调用 Java 服务器端特定的对象实例。
- 支持“工厂”模式，比如在使用了 EJB home 的情况下
- 这扩展了 JSON-RPC 协议，为 JSON-RPC JavaScript 客户端提供了动态代理创建的支持。



简单的 JSON-RPC-Java 例子

JSP 例子 (Hello.jsp)

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<jsp:useBean id="JSONRPCBridge" scope="session" class="com.metaparadigm.jsonrpc.JSONRPCBridge" />
<jsp:useBean id="hello" scope="session" class="com.metaparadigm.jsonrpc.test.Hello" />
<% JSONRPCBridge.registerObject("hello", hello); %>
<html>
<head>
<script type="text/javascript" src="jsonrpc.js"></script>
<script type="text/javascript" src="hello.js"></script>
</head>
<body bgcolor="#ffffff" onLoad="onLoad()">
<p>Who: <input type="text" id="who" size="30" value="Michael">
<input type="button" value="Say Hello" onclick="clickHello()"></p>
</body>
</html>
```

```
package com.metaparadigm.jsonrpc.test;

public class Hello
{
    public String sayHello(String who)
    {
        return "hello " + who;
    }
}
```

Java 例子 (Hello.java)

```
function onLoad()
{
    jsonrpc = new JSONRPCClient("/JSON-RPC");
}

function clickHello()
{
    var whoNode = document.getElementById("who");
    var result = jsonrpc.hello.sayHello(whoNode.value);
    alert("The server replied: " + result);
}
```

JavaScript 例子 (hello.js)



metaparadigm

进阶主题（异步调用）

异步调用与同步调用比较

- 同步调用可以使编程更方便
- 当调用在后台发生时，异步调用可以保持浏览器的活动状态（推荐采用）
- JSON-RPC-Java 使异步 RPC 调用更简单：





```
function gotAnimalResult(result, exception) {  
    if(exception) { alert(exception.message); }  
    // do stuff here ...  
}  
jsonrpc.country.getAnimals(gotAnimalResult, "Singapore");
```

- 赋予你的应用程序推送的能力
 - 产生一次异步调用接着服务器被这个调用所堵塞直到特定事件发生
- 支持多并发操作



进阶主题（类型映射）

Java 到 JavaScript 类型映射

-  允许 JSON-RPC-Java 透明的反序列化复杂嵌套的对象，并使用 Java 容器中的类，JSON-RPC 需要一种机制来保存类型信息。（译注：下面列举了障碍）
 - JavaScript 具有天生的类型无关性
 - 为容器通用而设计的 Java 单 Object 基类模式
（可以通过 Java 5.0 的参数化类型解决）
-  JSON-RPC-Java 通过给 JavaScript 传递对象时添加“javaClass”属性来实现“Class hinting”，以便当对象返回到 Java 端时提高序列化的准确性。
-  特殊的容器类型映射：List、Map、Set（详见开发文档）
-  Java Bean 可读性属性直接映射成了 JavaScript 对象（不用 get 或 set 前缀），当 JavaScript “bean” 对象传回 Java 端时，可写性属性也设置好了。



JSON-RPC-Java 演示

演示

 <http://oss.metaparadigm.com/jsonrpc/demos.html>

下载 (译注:)

 <http://oss.metaparadigm.com/jsonrpc-dist/json-rpc-java-0.8.tar.gz>

邮件列表

 <http://oss.metaparadigm.com/mailman/listinfo/json-rpc-java>

做贡献

```
# export CVSROOT=:pserver:anoncvs@cvs.metaparadigm.com:/cvsroot
# cvs login
Logging in to :pserver:anoncvs@cvs.metaparadigm.com:2401/cvsroot
CVS password: <enter 'anoncvs'>
# cvs co json-rpc-java
# cd json-rpc-java
< hack, hack, hack, ...>
# cvs diff -u | mail -s "my patch" michael@metaparadigm.com
```



metaparadigm