.2008/8/6

JSON-RPC FOR JAVA2.5 使用说明

《JavaScript 高级应用与实践》的延伸 | 夏天

JSON-RPC for Java 使用说明

夏天

2008

[QQ: 11602011] [轻量级、零入侵、级联调用 JSON-RPC for JAVA AJAX 框架]

目　　录

概述

继《JavaScript 高级应用与实践》(电子工业出版社.博文视点)之后推出的 json-rpc-for-java 开源代码，是仅仅 100 行的 javascript 代码和不到 10 个 java 文件实现的超级轻量级的通过 javaScript 快速调用 java 对象并返回任意对象的轻量级框架，并且支持级联调用，也就是说不需要额外 的 JavaScript 编程，就可以通过 javascript 调用被注册的 java 对象并返回 java 对象，如果被返回的对象还有方法，这个在 javascript 中返回的 java 对象的变量，还可以继续调用它的方法.....这就是这个轻量级 json-rpc-for-java 的神奇之处。

创意背景

发现其他的 JSON-RPC 开源的 JavaScript 代码太过繁杂，不够简洁，可维护性太差，而其注册复杂、配置繁多，使用不方便，并且不支持复杂对象的传入和级联调用功能。

应用前景和展望

该框架将封装即将发展多个扩展包：

1. **验证包**：用于常规的 Web 开发验证使用，比如身份证、邮件地址、电话号码、邮编、期号等；
2. **JavaScript 功能包**：包括 UI、功能的对象封装，其好处是不在是常规的将 JavaScript 代码下载到浏览器进行使用，而是按需加载，从而降低网络流量，提高网络的浏览速度。
3. **其他服务包**：包装 google 的在线翻译服务、图表服务等

术语

缩写	全称	描述
JSON	JavaScript Object Notation	JavaScript 对象的一种字面量描述格式，是一种轻量级的数据交换格式——相对于 XML，易于人阅读和编写，同时也易于机器解析
RPC	Remote procedure call	远程过程(函数、方法)调用
AJAX	asynchronous JavaScript and XML	狭义的解释是：异步的使用 XML 和 JavaScript 进行交互和通讯的一种技术；广义的而

		言，指一切 Web 上异步的技术
Java		一种编程语言
JavaScript		一种编程语言

链接

作者相关链接

作者 csdn 博客
作者新浪 600 多万次点击博客
作者网站

开源项目地址

http://code.google.com/p/json-rpc-for-java/

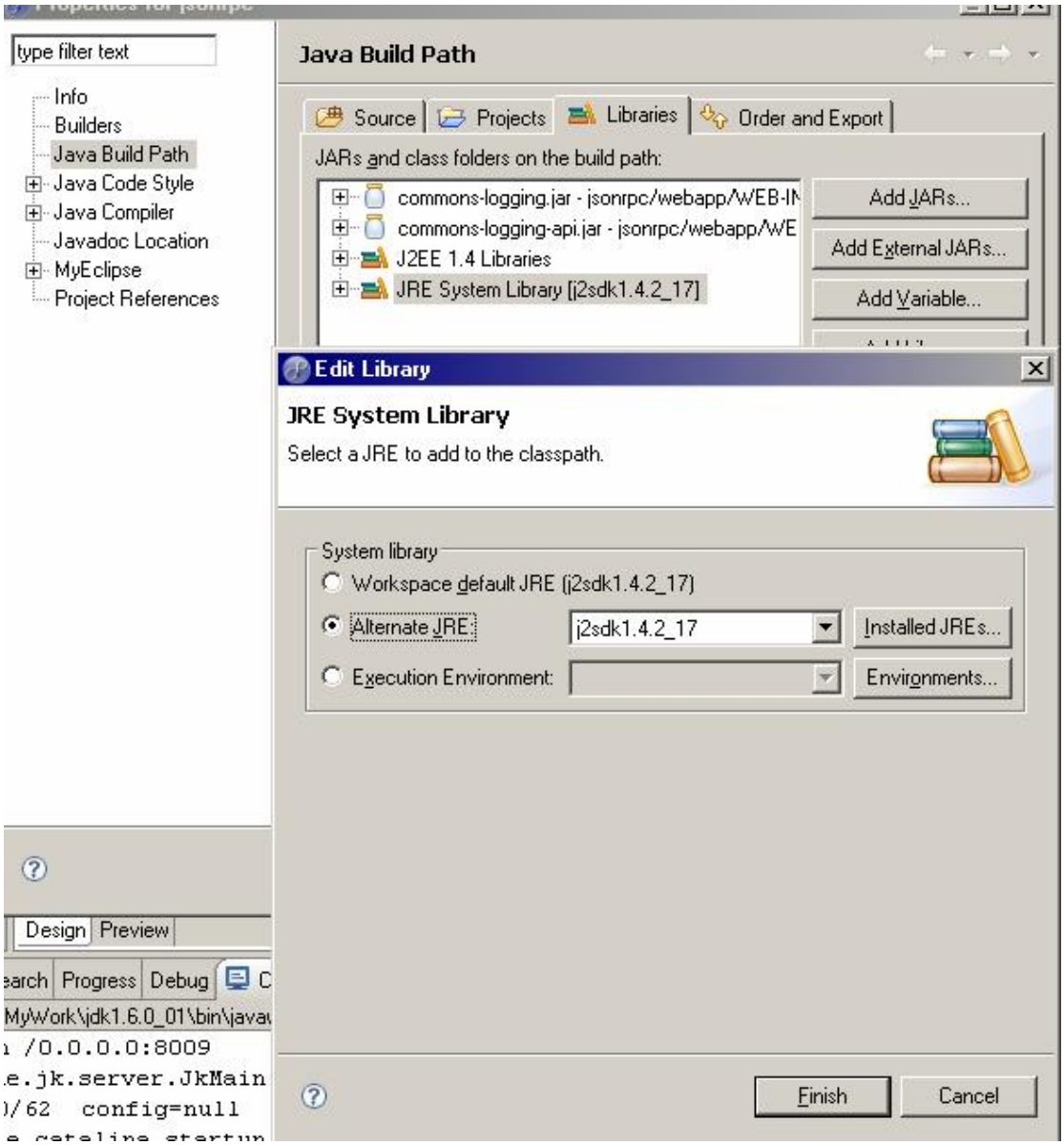
工程 svn 下载地址

http://json-rpc-for-java.googlecode.com/svn/trunk/
不需要用户名和密码。

示例工程下载地址

http://json-rpc-for-java.googlecode.com/files/JsonRpcExample2008-08-05.rar

测试环境：MyEclipse、JRE1.4(或 1.6)、tomcat 5.0(或 6.0) 如果你要测试，可以采用相应的环境，不一定要那么高版本的环境 ,Import 工程后请注意修改工程中 JRE 为正确的本版路径，如下图所示：



环境

支持的浏览器

IE4、IE5、IE6、IE7、IE8、>= FireFox 2.0、>= Opera、>= Safari、google 浏览器等等。力求设计与操作系统无关。

开发环境

Tomcat 6.0、Eclipse 3.2、JDK1.4。

运行环境

JRE1.4 或以上的 J2EE 环境。

同类产品分析比较

更加灵活的注册方式

和 DWR 相比，它更加灵活，既能够通过 web.xml 静态为所有访问者注册 JSON-RPC 服务对象，还能动态通过代码进行注册，并且所有调用的方法不需要事先声明和注册。

支持级联调用和复杂对象作为入参

区别与其他 JSON-RPC 框架的是，它支持级联调用，对 RPC 的 java 服务方法，还支持传入复杂如 Map、JavaBean 这样的对象。

参数

Java 服务方法入口参数类型

JavaScript 类型到 Java 类型的对照如下，如果入口参数为复杂对象，则要求**必须是能实例化的类型**：

JavaScript 对象	Java 对象	说明
String	java.lang.String	字符串
java.lang.Object	Java 复合类型，通常是 JavaBean 对象，或者是 java.util.Map	如果子元素不是简单类型，则遵循其他规则，否则以 String 处理，并且不支持深度嵌套的对象
number	java.util.Date java.sql.Timestamp	如果调用参数是日期类型，则自动以数字进行处理：new Date(Long.parseLong(szTmp01));
java.lang.Boolean	Blooean	对应的值：true、false
String	java.lang.Character	单引号的字符串，例如：‘c’
Number	java.lang.Short、 java.lang.Integer、 java.lang.Long、 java.lang.Float、 java.lang.Double、 java.math.BigDecimal	JavaScript 中都为数字对象，到参数里后自动转换类型

null	null	空对象

如果转换失败，入口参数将是一个没有数据的入口参数对象，如果转换发生异常，则以 null 传入。

Java 对象到 JavaScript 对象的对照表

调用异常时返回 false。

Java 对象	JavaScript 对象	说明
java.lang.String	String	
java.lang.Object	String	调用 java 对象的 toString()后转换到 JavaScript 里
java.util.Date、java.sql.Timestamp	String	格式为 yyyy-MM-dd HH:mm:ss.000，如果时分秒都为 0，则为：yyyy-MM-dd
java.lang.Boolean	Blooean	对应的值：true、false
java.lang.Character	String	单引号的字符串，例如：‘c’
java.lang.Short、java.lang.Integer、java.lang.Long、java.lang.Float、java.lang.Double、java.math.BigDecimal	Number	到 JavaScript 中都为数字对象，可以直接参与加、减、乘、除运算
java.util.Map	Object	例如：obj[“key1”]、obj[“key3”]、obj.key3，唯独没有以 function 作为属性的方法，当然，属于 Object.prototype 的 function 属性依然有的
java.util.List	Array	例如：a[0]、a[2].getList() 也就是说 List 里也可以存在复合对象，这些对象依然可以有自己的方法
null	null	空对象
其他 Java 对象	Object	例如：obj.displayName()、obj.aac001，可以有属性和方法

返回类型：

实现接口：

jcore.jsonrpc.common.face.IResultObject

这样，在 JavaScript 中不用再调用 getErrMsg，直接使用你存放异常消息的变量名就可以获得异常、错误消息。

实例化：

或者在你的异步服务方法中实例化：

```
jcore.jsonrpc.common.ResultObject
```

调用 `ResultObject.setResult` 将你实际返回的对象 set 进去，这样，当发生异常消息的时候可以在 js 中用如下的方式获取

```
// 错误消息，和结果一次性取出，而不用调用 getErrMsg
// 因为当调用 getErrMsg 的时候就会再次发起异步请求
var szMsg = rpcRstObj['errMsg'],
// 这是返回的对象
oRst = rpcRstObj['result']
;
```

注意事项

如果你的 java 服务对象返回的是 Object、Bean、Map 或者自定义对象，不能有属性 _name_、_id_，这两个属性被本框架内部使用；

功能介绍

自动捕获异常

在你编写的 java 服务类的方法中不需要 try{….}catch(Exception e){}，本框架会为你捕获异常错误消息，当你在 javascript 中没有获取到正确的数据，可以调用异步对象的方法 getErrMsg() 获取异常消息，该方法封装在 jcore.jsonrpc.common.JsonRpcObject 中，也就是 AJAX 服务 java 基类中。

当返回类型为：

`jcore.jsonrpc.common.ResultObject`

jcore.jsonrpc.common.face.IResultObject

则可以只用通过返回的对象中获取 oRst['errMsg']异常消息。

JavaScript 中释放注册的 Java 服务对象

你只需要在 JavaScript 中调用 release（）就可以释放注册的 Java 对象资源，详细见示例工程，或者见：

http://code.google.com/p/json-rpc-for-java/wiki/Wiki32

级联调用功能

不明白的地方请结合示例工程进行理解。

1、Java 中注册复合对象 myjsonrpc

2、JSP JavaScript 中获取该对象：var myjsonrpc = JsonRpcClient().myjsonrpc;

3、调用被注册的 java 对象的方法 getMyObj，返回复合的 java 对象 TestDomain：

```
var oDomain = myjsonrpc. getMyObj();
// 继续调用该返回的 java 对象的方法
alert(oDomain. toXml())
```

或者：alert(myjsonrpc. getList()[1].toXml());

如果 toXml 返回的还是一个复合的 Java 对象，你可以继续在 JavaScript 中继续调用，而不需要额外的编程。

按需加载 JavaScript 库

Base

方法名	说明	使用举例

A(a)	转换有 length 属性的对象为 Array。将参数 a 对象转换为有效的 Array 对象；a 参数必须为有效的拥有 length 属性的对象	var json = JsonRpcClient("JRPC");
		var o = json.LoadJsObj("Base");
		o.id("myDivId");// 获取对象
		// 自动保存滚动条的位置
		o.autoSaveScroll("myTextArea");

id(s)	根据 id 获取对象。如果参数 s 为 String，则以他为 id 获取对象并返回，否则返回 s	
addEvent(o,t,f)	给对象绑定事件处理。 o：为 HTML 对象 t：为事件名，例如 click、focus f：为事件处理函数	
getCookie(k)	获取名字为 k 的 cookie 值	
setCookie(k,v)	将名字为 k，值为 v 的键值对设置到 cookie；如果 v 为无效的值，则删除该项 cookie	
autoSaveScroll(o)	设置对象 o 为自动保存滚动条位置——人性化	

	设计	
clearScroll(o)	通常，在提交数据后不需要保持对象的滚动条位置，因此需要调用该方法在提交数据前清楚滚动条存储信息	

使用

Web.xml 配置

需要在 web.xml 中加入下面的配置

```
<servlet>
        <servlet-name>JSONRPCServlet</servlet-name>
        <servlet-class>
            jcore.jsonrpc.servlet.JSONRPCServlet
        </servlet-class>

<!—
这里注册的对象将对所有访问者起作用
<init-param>
            <param-name>regAppClassNames</param-name>
            <param-value>
        注册这些对象，让所有的web用户都能使用这些JSON-RPC java对象
        这些对象会复制到每个用户的session对象中，因此他们必须实现接口Serializable
        以分号作为分割符号
                myTest:jcore.jsonrpc.test.Test;
                myTrs:jcore.jsonrpc.test.A
            </param-value>
        </init-param>
-->

        <load-on-startup>2</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>JSONRPCServlet</servlet-name>
        <url-pattern>/JRPC</url-pattern>
    </servlet-mapping>
```

引入 Jar 包

需要在工程中引入：JSON-RPC.jar、commons-logging.jar、commons-logging-api.jar，其中后面两个 jar 在示例工程中的 JsonRpcExample\webapp\WEB-INF\lib\ 下。示例工程下载地址：

http://json-rpc-for-java.googlecode.com/files/JsonRpcExample2008-08-05.rar

而，JSON-RPC.jar，你也可以引入源代码重新进行打包。

AJAX 服务 Java 类的编写

JsonRpcObject 基类中的方法列表

方法名	说明
getErrMsg	在 javaScript 中调用获取异常、错误消息使用
getRequest	在 java 服务的 **jcore.jsonrpc.common.JsonRpcObject** 继承子类中使用，获取到 request 对象，以便获取 session 等对象
setErrMsg	同上，在继承的子类中设置错误或异常消息的方法；该方法是框架抓取异常消息填写的方法
setRequest	框架系统使用，注入 request 的地方

服务类示例

必须继承与 **jcore.jsonrpc.common.JsonRpcObject**，并实现接口 **java.io.Serializable**。

例如示例工程中的 AJAX 服务 Java 类：

```
package test;
import java.io.Serializable;
import java.util.ArrayList;
import java.util.Date;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import jcore.jsonrpc.common.JsonRpcObject;

public class TestObject extends JsonRpcObject implements Serializable{
```

```
    private static final long serialVersionUID = 1L;

    private List myList = new ArrayList();
    private Map map = new HashMap();

    public TestObject()
    {
        myList.add("good");
        myList.add(new TestDomain());
        // map中也可以放入复合对象
        map.put("first", "第一条值");
        map.put("p2", new Date());
        map.put("domain", myList.get(1));
    }

    /***
     * 返回Map对象
     * @return
     */
    public Map getMap()
    {
        return map;
    }

    /***
     * 获取一个普通对象
     * @return
     */
    public Object getStr()
    {
        return myList.get(0);
    }

    /***
     * 获取一个复合对象
     * @return
     */
    public Object getMyObj()
    {
        return myList.get(1);
    }

    /***
     * 获取List对象
```

```
     * @return
     */
    public List getList()
    {
        return myList;
    }
}
```

自己基类的编写

同样，你可以继承 **jcore.jsonrpc.common.JsonRpcObject** 实现一些基类，这样在自己的项目中更加方便实用，例如：

```
package com.yinhai.yhsi2.web.common;

import com.yinhai.webframework.session.UserSession;
import jcore.jsonrpc.common.JsonRpcObject;

public abstract class Yhsi2JsonRpcObj extends JsonRpcObject {

    private UserSession us = null;
    public Yhsi2JsonRpcObj() {
        super();
    }

    public UserSession getUs() {
        if(null == us)
            us = UserSession.getUserSession(getRequest());
        return us;
    }
}
```

注意事项

RPC 的 Java 服务类返回的对象不能有成员变量引用自身对象，服务类对象不能有成员变量为 Service 或 BPO 对象，因为这些对象内部通常有一个成员变量引用了自身，这会导致服务类无法正常转换为正确的 JavaScript 对等的应用类，会导致堆栈溢出的控制台异常错误信息——我们预计在下一版本中增强这样的自身引用的检测并加以回避。

AJAX 服务 Java 类的注册

// 注意，被注册的类必须是能被实例化的类

jcore.jsonrpc.common.JsonRpcRegister.registerObject(us, "myjsonrpc", test.TestObject.class);

使用 test.TestObject.class 的方式是保证多次注册不至于 test.TestObject 被多次注册而执行多次实例化，从而提高性能，并允许多次注册——实际上内部只注册了一次。

自己注册基类的编写

当然，你也可以继承 jcore.jsonrpc.common.JsonRpcRegister 以便使得在应用菜单切换的时候释放资源，例如：

```
package com.yinhai.yhsi2.web.common;
import javax.servlet.http.HttpServletRequest;

import jcore.jsonrpc.common.Content;
import jcore.jsonrpc.common.JSONRPCBridge;

import com.yinhai.webframework.session.UserSession;
import jcore.jsonrpc.common.JsonRpcRegister;

/***
 * 注册 JsonRpc 对象
 * @author just
 *
 */
public class JsonRpcRegister extends jcore.jsonrpc.common.JsonRpcRegister{

	/***
	 * 通过 request 来注册对象
	 * @param request
	 * @param szKeyName
	 * @param o
	 */
	public static void registerObject(HttpServletRequest request, String szKeyName, Object o)
	{
		registerObject(UserSession.getUserSession(request), szKeyName, o);
	}

	/***
	 * 通过 request 来注册对象
```

```
     * @param request
     * @param szKeyName
     * @param o
     */
    public static void registerObject(UserSession us, String szKeyName, Object o)
    {
        if(null != us)
        {
            JSONRPCBridge brg =
(JSONRPCBridge)us.getCurrentBusiness().getSessionResource(Content.RegSessionJSONRPCNa
me);
            // 如果是第一次就注册对象
            if(null == brg)

us.getCurrentBusiness().putSessionResource(Content.RegSessionJSONRPCName, brg = new
JSONRPCBridge().setSession(us.getHttpSession()));
            brg.registerObject(szKeyName, o);
        }
    }



    /***
     * 通过 request 来注册对象
     * @param request
     * @param szKeyName
     * @param o
     */
    public static void registerObject(HttpServletRequest request, String szKeyName, Class o)
    {
        registerObject(UserSession.getUserSession(request), szKeyName, o);
    }

    /***
     * 通过 request 来注册对象
     * @param request
     * @param szKeyName
     * @param o
     */
    public static void registerObject(UserSession us, String szKeyName, Class o)
    {
        if(null != us)
        {
            JSONRPCBridge brg =
```

```
(JSONRPCBridge)us.getCurrentBusiness().getSessionResource(Content.RegSessionJSONRPCNa
me);
                    // 如果是第一次就注册对象
                     if(null == brg)

us.getCurrentBusiness().putSessionResource(Content.RegSessionJSONRPCName, brg = new
JSONRPCBridge().setSession(us.getHttpSession()));
                     try {
                          brg.registerObject(szKeyName, o.newInstance());
                    } catch (InstantiationException e) {
                    } catch (IllegalAccessException e) {
                    }
               }
      }
}
```

银海公司专用服务类示例

```
package com.yinhai.yhcip.common;

import java.util.Map;

import jcore.jsonrpc.common.JsonRpcObject;

import com.yinhai.sysframework.persistence.IDao;
import com.yinhai.sysframework.service.ServiceLocator;

/**
 * Rpc 调用
 * @author just
 *
 */
public class Rpcyhcip extends JsonRpcObject{

      /**
       * 远程异步调用过程
       * @param szDaoId   dao 名称 id
       * @param szSqlId   sql 语句 id
       * @param map          入口参数
       * @return
       */
      public boolean delete(String szDaoId, String szSqlId, Map map)
```

```
    {
        try
        {
                // 如果需要 UserSession 对象：UserSession us =
UserSession.getUserSession(this.getRequest());
                // getRequest 方法是父类中实现
            IDao dao =(IDao) ServiceLocator.getInstance().getService(szDaoId);
            return 0 < dao.delete(szSqlId, map);
        }catch(Exception e)
        {
            e.printStackTrace();
            setErrMsg(e.getMessage());
        }
        return false;
    }
}
```

JSP 中的使用

引入 JsonRpcClient.js

```
<script charset="UTF-8" type="text/JavaScript" src="JsonRpcClient.js"></script>
```

最新的文件可以从 http://code.google.com/p/json-rpc-for-java/ 下载

调用

```
<script charset="UTF-8" type="text/JavaScript"><!--//--><![CDATA[//><!--

// myjsonrpc 就是通过 JsonRpcRegister.registerObject 注册的名字
// 这时候这里的 rpc 就拥有了通过 JsonRpcRegister.registerObject 注册的
// 异步对象的相应方法了
var rpc = JsonRpcClient().myjsonrpc;
// 传入个人编号获取人的基本信息并填充到界面上
if("dto(aac001)" === o.name && 0 === "aab001".getValue().length)
  {
      if(0 < o.value.length)
      {
```

```
        // 获取到的 myjsonrpc 同样有 aac001,aab001 等等属性，
        // 你可以直接使用，同样有 getAac001()等方法，
        // 可以直接使用，而不需要额外的编码
            var myjsonrpc = rpc.getEmployeeBaseInfo(o.value), errMsg = rpc.getErrMsg();
            if(0 < errMsg.length)
                return o.focus(), alert(errMsg), false;
            for(var k in myjsonrpc)
              if(6 === k.length && myjsonrpc[k])
                k.getObj() && k.setValue(myjsonrpc[k]);
            fnSetAtbt("dto(aac001)".getObj(),1),
            fnSetAtbt("dto(aab001)".getObj(),4),
            "aab001".focus();
            // 关闭错误消息提示
            ;
        }
    }
// 2008-08-02 增加自动拦截异常消息功能，因此在你写的代码中不需要编写 try catch
// 如果有异常消息，可以在 js 中调用 rpc.getErrMsg()获得
// 如果需要释放被注册名为 myjsonrpc 的对象 JsonRpcObj，可以在 js 中调用 rpc.release();
// 2008-08-13 增加对 FireFox 3.01 的支持，增加传入 JavaBean、Map 参数的支持
//--><![]></script>
```

如果本调用的方法 getEmployeeBaseInfo 的第一个参数是 function，则把它作为异步的回调函数。

调用未注册和配置的类方法

1、 首先，被调用的类需要继承 jcore.jsonrpc.common.JsonRpcObject 或实现接口 jcore.jsonrpc.common.face.IjsonRpcObject，并有默认的构造函数；

例如：

```
package test.rpc;
import jcore.jsonrpc.common.JsonRpcObject;
public class MyTestRpc extends JsonRpcObject {
    /**
     * 调用：rpc.getRpcObj('test.rpc.MyTestRpc').getTestMsg()
     * @return
     */
    public String getTestMsg()
    {
        return "噢，成功了！";
    }
}
```

2、 JSP 的 JavaScript 中调用的方式，例如：

```
alert(rpc.getRpcObj('test.rpc.MyTestRpc').getTestMsg());
```