

2008/8/6

忍

JSON-RPC FOR JAVA 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

JSON-RPC for Java 使用说

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

目 录

概述.....	4
链接.....	4
作者相关链接.....	4
开源项目地址.....	4
工程 svn 下载地址	4
示例工程下载地址.....	4
支持的浏览器.....	5
Java 对象到 JavaScript 对象的对照表.....	5
功能介绍.....	6
自动捕获异常.....	6
JavaScript 中释放注册的 Java 服务对象.....	6
级联调用功能.....	7
使用.....	7
Web.xml 配置	7
引入 Jar 包.....	7
AJAX 服务 Java 类的编写	8
自己基类的编写.....	9
AJAX 服务 Java 类的注册	10
自己注册基类的编写.....	10
JSP 中的使用	12
引入 JsonRpcClient.js.....	12
调用.....	12

概述

继《JavaScript 高级应用与实践》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。

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

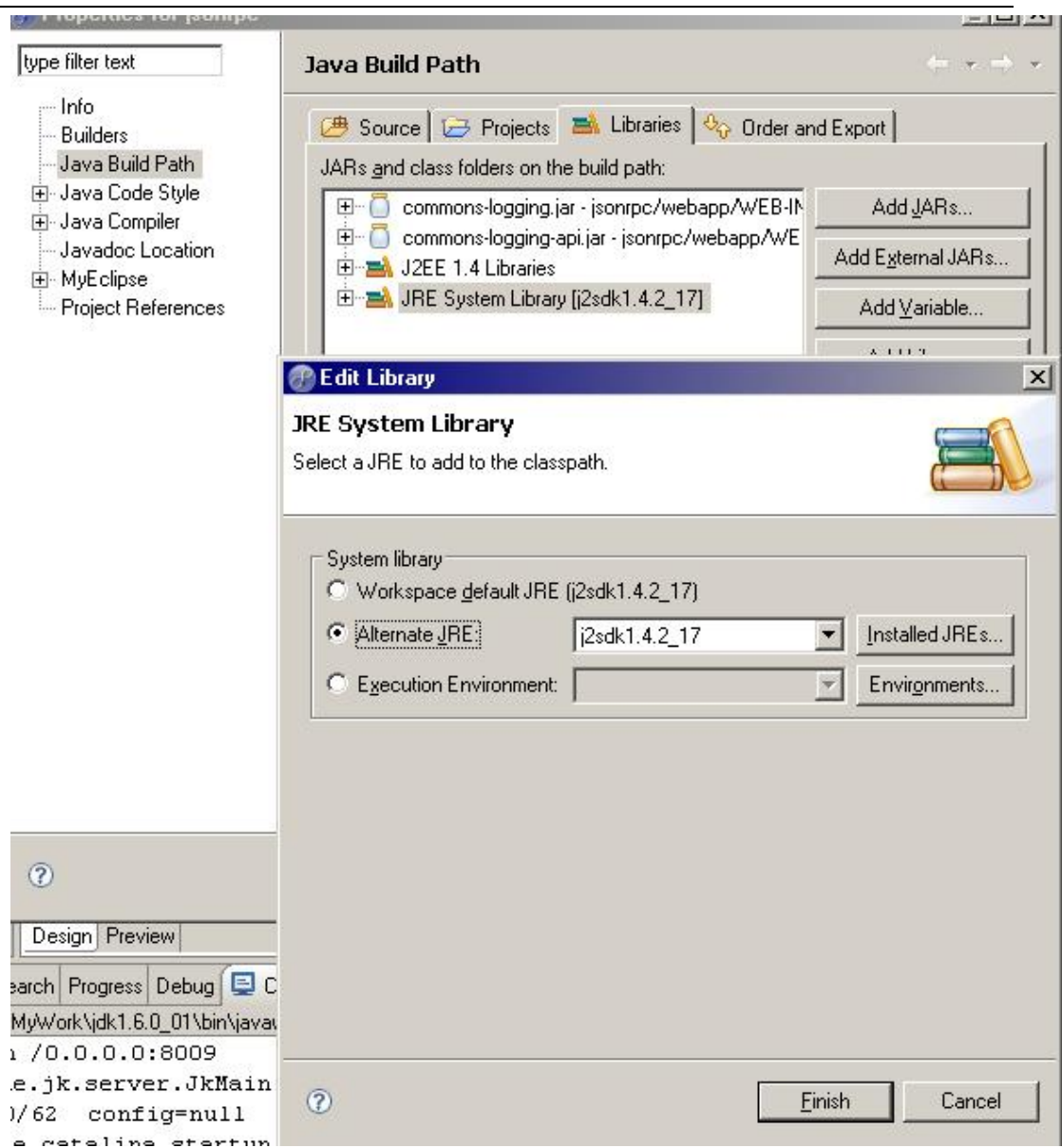
<http://json-rpc-for-java.googlecode.com/svn/trunk/>

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 为正确的路径：



支持的浏览器

IE4、IE5、IE6、IE7、IE8、 FireFox、Opera、Safari 等等。

Java 对象到 JavaScript 对象的对照表

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	可是为 yyyy-MM-dd HH:mm:ss.000，如果时分秒都为 0，则为：yyyy-MM-dd
java.lang.Boolean	Bloolean	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 function 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

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){}`，本框架会为你捕获错误消息，但你在 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。

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。

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>
    <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

必须继承与 `jcrc.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 jcrc.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;  
    }  
}
```

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.RegSessionJSONRPCName);
            // 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.RegSessionJSONRPCName);
        // 如果是第一次就注册对象
        if(null == brg)
            us.getCurrentBusiness().putSessionResource(Content.RegSessionJSONRPCName, brg =
new JSONRPCBridge().setSession(us.getSession()));
        try {
            brg.registerObject(szKeyName, o.newInstance());
        } catch (InstantiationException e) {
        } catch (IllegalAccessException e) {
        }
    }
}
}
```

JSP 中的使用

引入 `JsonRpcClient.js`

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

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();

//--><![ ]></script>
```