目录

[目的 1](#_Toc301887817)

[设计目标 1](#_Toc301887818)

[便于测试 1](#_Toc301887819)

[易于使用 1](#_Toc301887820)

[跨语言调用 1](#_Toc301887821)

[错误检测 1](#_Toc301887822)

[权限控制 2](#_Toc301887823)

[协议 2](#_Toc301887824)

[请求 2](#_Toc301887825)

[响应 2](#_Toc301887826)

[json 2](#_Toc301887827)

[xml 3](#_Toc301887828)

[主要类描述 3](#_Toc301887829)

[API使用设计 3](#_Toc301887830)

[服务端服务发布 3](#_Toc301887831)

[客户端服务使用 4](#_Toc301887832)

[参考 5](#_Toc301887833)

# 目的

描述RPC的具体API使用规范，以便统一现在基于http的远程RPC调用。并提供具体实现原理描述，以便了解整体架构

|  |  |  |
| --- | --- | --- |
| 日期 | 作者 | 说明 |
| 2011/8/23 | 邱百超 | 草稿 |
|  |  |  |
|  |  |  |

# 设计目标

## 便于测试

1．基于HTTP,支持get及post接口

* **Get请求：**便于测试使用，以便于直接使用apache ab进行压力测试
* **Post请求:** 实际的远程RPC调用全部使用Post请求

2. 服务端提供生成URL访问列表

* 列举出所有的服务列表
* 列举出所有服务的方法列表

## 易于使用

生成server端动态代理，减少客户端开发工作量

生成client动态代理，减少客户端开发工作量

## 跨语言调用

使用json及xml作为通讯协议，便于不同语言的客户端调用

## 错误检测

提供服务接口的快速失败检测机制

## 权限控制

提供基于客户端IP地址的权限访问过滤

# 协议

## 请求

**提供两种URL参数传递方式**

1. 基于参数位置的URL传递

/rpc/BlogService?\_id=123&\_method=userinfo&\_params=badqiu,25,F

1. 基于参数名称的URL传递

/rpc/BlogService?\_id=123&\_method=userinfo&username=123&password=456

**技术限制**

由于在java代码中，是无法直接得到一个参数的方法名称的，必须通过解析class字节码或者是使用paranamer类库来得到方法的参数名称。所以如果要使用命名参数的实现远程RPC调用，必须通过其它技术手段配合才行。

**请求参数含义:**

\_id : 请求ID，用于异步调用使用，跟response对应起来

\_method : 请求调用的方法

\_params : 无序的命名参数调用

\_format : xml,json

## 响应

error\_code: 错误码,可以是数字或者是字符串

error: 具体的错误描述

result: 具体的result对象

### json

{

error\_code: ‘error\_code’,

error: ’error message’,

result: { }

}

### xml

<ret>

<error\_code>error code</error\_code>

<error >error message</error>

<result>result object</result>

</ret>

# 主要类描述

WebServiceException: 远程调用如果响应结果异常，则抛出此异常

RPCExporter: 服务端用户发布类,用于将一个接口发布成WebService

RPCProxyFactoryBean: 客户端服务代理生成类，用于生成一个动态代理，以便客户端使用

# API使用设计

## 服务端服务发布

**发布方式1(无序参数)**

<bean class=”com.duowan.rpc.simple.RPCExporter”>

<property name=”serviceInterface”>com.duowan.hell.EchoWS</perperty>

<property name=”service” ref=”echoWSImpl”/>

</bean>

**发布方式2(命名参数)**

@RequestMapping("/EchoService")

**public** **class** EchoController {

@Autowired

**private** EchoService echoService;

@RequestMapping(params="\_method=say")

**public** **void** say(String name,HttpServletResponse response) {

**try** {

Object result = echoService.say(name);

outputResult(result,response);

}**catch**(Exception e) {

outputError(e,response);

}

}

}

## 客户端服务使用

**客户端使用方式1(无序参数):**

<bean id=”” class=”com.duowan.rpc.simple.RPCProxyFactoryBean”>

<property name=”serviceInterface”>com.duowan.hell.Echo</perperty>

<property name=”serviceUrl” value=”http://echo.game.yy.com:8080/simplerpc/EchoWS”/>

<property name=”connectTimeout” value=”60000”/>

<property name=”invokeTimeout” value=”60000”/>

</bean>

**客户端使用方式2(命名参数)**

**public** **class** OnlineTimeWebServiceImpl **implements** OnlineTimeWebService {

**private** String url;

**public** OnlineTimeWebServiceImpl(String url) {

**this**.url = url;

}

**public** List<OnlineTime> listAllOnlineTime(String product,

String passport) {

Map<String,Object> params = **new** HashMap<String,Object>();

params.put("product", product);

params.put("passport", passport);

JsonResult result = JsonHttpRPCUtil.*readUrlForObject*(url,"listAllOnlineTime",params, JsonResult.**class**);

**if**(result == **null**)**throw** **new** RuntimeException("请求的地址不存在"+url);

**if**(StringUtils.isEmpty(result.getErrorCode()))

**return** JsonUtil.*toListObject*(result.getContent(), OnlineTime.**class**);

**else**

**throw** **new** RuntimeException(result.getErrorCode(),result.getError());

}

}

# 参考

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

<http://paranamer.codehaus.org/> paranamer