

Generic调用说明

被章耿添加，被武美君最后更新于一月 06, 2017

在网关等场景下，调用者是拿不到服务端的class或者jar包，那此时发起调用，该如何处理？

JSF支持泛化调用，只需要指定接口、方法、参数类型、参数值，就可以完成调用。

GenericService 接口如下：

com.jd.saf.gd.GenericService

```
package com.jd.jsf.gd;
import com.jd.jsf.gd.msg.ResponseListener;
public interface GenericService {
    /**
     * 泛化调用
     *
     * @param method
     *      方法名
     * @param parameterTypes
     *      参数类型
     * @param args
     *      参数列表
     * @return 返回值
     */
    public Object $invoke(String method, String[] parameterTypes, Object[] args);
    /**
     * 异步回调的泛化调用
     *
     * @param method
     *      方法名
     * @param parameterTypes
     *      参数类型
     * @param args
     *      参数列表
     * @param listener
     *      结果listener
     */
    public void $asyncInvoke(String method, String[] parameterTypes, Object[] args, ResponseListener listener);
}
```

PS：参数说明（具体使用在更下面）

String[] parameterTypes: 要调用的方法的参数类型，
Object[] args: 参数列表【包括值及自定义类型等信息】

示例：

调用方法：public String echoMultipleParam(String str, Integer data);
\$invoke("echoMultipleParam", new String[]{"java.lang.String", "java.lang.Integer"}, new Object[]{"maggie", 1});

调用方法：public byte[] randomSizeResult();
\$invoke("randomSizeResult", null, null);

调用方法：public String getTestObject(TestObject object);
Map map = new HashMap();
map.put("name", "1");
map.put("type", "YELLOW"); // EnumType.YELLOW
map.put("values", list);
map.put("class", "com.jd.testjsf.vo.TestObject"); //调用时如果传递是自定义对象，且没有自定义对象的class，可以通过hashmap进行传递，设置一个“class”的key。
\$invoke("getTestObject", new String[]{"com.jd.testjsf.vo.TestObject"}, new Object[]{map});


返回对象说明：

Object：调用方法的实际返回统一Object返回。如果服务端方法返回的是自定义的POJO，则客户端可以将Object转成JSON进行相应业务处理。//JSON.toJSON(object)

使用方式如下（Generic调用多用于API方式，所以就举API的例子）：


1、Provider发布服务的时候照常启动，无需增加任何配置

2、Consumer调用的时候，只需要配置generic="true"，

 **ConsumerConfig.refer()**得到泛化调用实例的操作很重，包括连接注册中心，获取配置，获取服务列表，建立长连接，生成代理类等一系列动作，

使用请缓存ConsumerConfig对象（推荐）或者ConsumerConfig.refer()后的GenericService代理类对象！！！！

如果是网关建议使用 接口名+alias+protocol 作为Key，ConsumerConfig作为Value。

 调用时如果传递是自定义对象，且没有自定义对象的class，可以通过hashmap进行传递，设置一个“class”的key。

同步调用和普通异步调用参考如下：

GenericClientMainAPI

```

package com.jd.testjsf.generic;
import java.util.ArrayList;
import java.util.List;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.core.SpringVersion;
import com.jd.jsf.gd.GenericService;
import com.jd.jsf.gd.config.ConsumerConfig;
import com.jd.jsf.gd.config.RegistryConfig;

public class GenericClientMainAPI {
    private final static Logger logger = LoggerFactory.getLogger(GenericClientMainAPI.class);

    public static void main(String[] args) {
        // 注册中心实现（必须）
        RegistryConfig jsfRegistry = new RegistryConfig();
        jsfRegistry.setIndex("i.jsf.jd.com"); // 测试环境192.168.150.121 i.jsf.jd.com
        LOGGER.info("实例RegistryConfig");
        // 服务消费者连接注册中心，设置属性
        ConsumerConfig<GenericService> consumerConfig = new ConsumerConfig<GenericService>();
        consumerConfig.setInterfaceId("com.jd.testjsf.HelloService");// 这里写真实的类名
        consumerConfig.setRegistry(jsfRegistry);
        consumerConfig.setProtocol("jsf");
        consumerConfig.setAlias("JSF_0.0.1");
        consumerConfig.setGeneric(true); // 需要指定是Generic调用true
        // consumerConfig.setAsync(true); // 如果异步
        LOGGER.info("实例ConsumerConfig");
        // 得到泛化调用实例，此操作很重，请缓存consumerConfig或者service对象！！！！（用map或者全局变量）
        GenericService service = consumerConfig.refer();
        while (true) {
            try {
                // 传入方法名，参数类型，参数值
                Object result = service.$invoke("echoStr", new String[]{"java.lang.String"},
                    new Object[]{"zhanggeng"});
                // 如果异步
                // ResponseFuture future = RpcContext.getContext().getFuture();
                // result = future.get();
                LOGGER.info("result :{}", result);

                // 如果传递对象的，如果没有对象类，可以通过map来描述一个对象
                Map map = new HashMap();
                map.put("id", 1);
                map.put("name", "zhangg21genericobj");
                map.put("class", "com.jd.testjsf.ExampleObj"); // class属性就传真实类名
                // 或者用json转map，而不是一个个put
                // map = JSON.parseObject("
                {"id":1,"name":"zzzzzz","class":"com.jd.testjsf.ExampleObj", Map.class});
                Object objresult = service.$invoke("echoObject", new String[]{"com.jd.testjsf.ExampleObj"},
                    new Object[]{map});
                LOGGER.info("obj result :{}", objresult);

                //如果参数传递list对象。接口方法是：public List<ExampleObj> echoObjectList(List<ExampleObj>
list)
                List list = new ArrayList();
                Map map1 = new HashMap();
                map1.put("id", 1);
                map1.put("name", "polly");
                map1.put("class", "com.jd.testjsf.ExampleObj");
                list.add(map1);
                Object result1 = service.$invoke("echoObjectList", new String[]{"java.util.List"}, new
Object[]{list});
                LOGGER.info("obj result :{}", result1);

                /**
                 * public String getTestObject(TestObject testObject)
                 * 方法名：getTestObject
                 * 入参：TestObject，自定义POJO,含枚举类型、List<Domain>
                 * 返回：String
                 */
                List list = new ArrayList<>();
                for(int j=0 ; j < 10 ; j++){
                    Map domain = new HashMap();
                    domain.put("name", "polly");
                    domain.put("sex", "girl");
                    domain.put("class", "com.jd.testjsf.vo.Domain");

```

```

        list.add(domain);
    }
    Map map = new HashMap();
    map.put("name", "1");
    map.put("type", "YELLOW");// EnumType
    map.put("values", list);//List<Domain>
    map.put("class", "com.jd.testjsf.vo.TestObject");
    Object message = service.$invoke("getTestObject", new String[]
{"com.jd.testjsf.vo.TestObject"},
        new Object[]{map});
    LOGGER.info("response msg from server :{}",message );

    /**
     * public List<String> changeColorType(String str,Set<ColorType> colorTypes);
     * 方法名: changeColorType
     * 入参: String,Set<ColorType>
     * 返回:List<String>
     */
    Set set = new HashSet();
    set.add("BLANK");//自定义枚举
    set.add("GREEN");//自定义枚举
    set.add("RED");//自定义枚举
    Object listObject = service.$invoke("changeColorType", new String[]{"java.lang.String",
"java.util.Set"},
        new Object[]{"hello", set});
    LOGGER.info("response listObject from server :"+listObject );//response listObject from
server :[BLANK, GREEN, RED]

        } catch (Exception e) {
            LOGGER.error("", e);
        }
        try {
            Thread.sleep(3000);
        } catch (Exception e) {
        }
    }
}
}

```

异步回调的参考如下:

```

package com.jd.testjsf.async;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import com.jd.jsf.gd.GenericService;
import com.jd.jsf.gd.config.ConsumerConfig;
import com.jd.jsf.gd.config.RegistryConfig;
import com.jd.jsf.gd.msg.ResponseListener;

public class AsyncGenericClientMainAPI {
    private final static Logger LOGGER = LoggerFactory.getLogger(AsyncGenericClientMainAPI.class);
    static int msgid = 0;
    public static void main(String[] args) {
        // 注册中心实现（必须）
        RegistryConfig jsfRegistry = new RegistryConfig();
        jsfRegistry.setIndex("i.jsf.jd.com"); // 测试环境192.168.150.121 i.jsf.jd.com
        LOGGER.info("实例RegistryConfig");
        // 服务提供者连接注册中心，设置属性
        ConsumerConfig<GenericService> consumerConfig = new ConsumerConfig<GenericService>();
        consumerConfig.setInterfaceId("com.jd.testjsf.HelloService");// 这里写真实的类名
        consumerConfig.setRegistry(jsfRegistry);
        consumerConfig.setProtocol("jsf");
        consumerConfig.setAlias("JSF_0.0.1");
        consumerConfig.setGeneric(true); // 需要指定是Generic调用true
        LOGGER.info("实例ConsumerConfig");
        // 得到泛化调用实例，此操作很重，请缓存consumerConfig或者service对象！！！！（用map或者全局变量）
        GenericService service = consumerConfig.refer();
        while (true) {
            try {
                // 传递对象的，如果没有对象类，可以通过map来描述一个对象
                Map map = new HashMap();
                map.put("id", 1);
                map.put("name", "zhangg21genericobj");
                map.put("class", "com.jd.testjsf.ExampleObj"); // class属性就传真实类名
                // 或者用json转map，而不是一个个put
                // map = JSON.parseObject("
                {"id":1,"name":"zzzzzz","class":"com.jd.testjsf.ExampleObj"}");
                // 异步回调（控制发送速度，发太快小心客户端自己内存爆了）
                service.$asyncInvoke("echoObject", new String[]{"com.jd.testjsf.ExampleObj"},
                    new Object[]{map}, new ResponseListener() {
                        @Override
                        public void handleResult(Object result) {
                            LOGGER.info("[{}] aysnc invoke result :{}", msgid++, result);
                        }
                        @Override
                        public void catchException(Throwable e) {
                            LOGGER.error("[{}] aysnc invoke error :{}", msgid++, e.getMessage());
                        }
                    });
            } catch (Exception e) {
                LOGGER.error("echoObject出现异常", e);
            }
            try {
                Thread.sleep(2000);
            } catch (InterruptedException e) {
            }
        }
    }
}

```