SAF TO JSF

工具

被章耿添加，被章耿最后更新于一月 06, 2016

- 1. 主要区别
  - 1.1 区别之一：序列化协议
  - 1.2 区别之二：配置
  - 1.3 区别之三 ：关于方法重载
- 2. 服务端先升级到JSF
- 3. 客户端先升级到JSF

联系方式： 优先发组邮箱 **jsf@jd.com** **SAF小组成员**

JSF和SAF是独立的框架，可以同时存在，互不影响。 参见JSF与SAF的兼容与比较

这里主要说明升级的方式：

1. 主要区别

1.1 区别之一：序列化协议

注意：**jsf**的默认序列化方式从 **hessian** 换成了 **msgpack** **JSF序列化说明（MsgPack）**

两者的区别：

hessian序列化的时候，会写入字段名称，然后字段值，你可以想象为一个map。
msgpack序列化的时候，不写入字段名字，会按字段顺序写入值，你可以想象为一个数组。

从这就可以看出：

1. hessian产生的数据包较大，msgpack产生的数据包较小。网络传输数据更小。
2. 序列化中hessian的性能较差，（相当于每次map按名字取值）
   msgpack性能更佳，（相当于数组取值）
   压测结果不同场景显示提高10% - 30%。（从数组取值比map高效）
3. hessian的扩展性更好，上下兼容时，可以随意添加字段位置（相当于map可以随便赋值）
   msgpack的性能更佳，上下兼容时，需要保证字段顺序（包括枚举顺序）。 参见 JSF FAQ#22.JSF默认的Msgpack序列化，接口对象里增减字段如何处理？
4. 其它一些差异：例如hessian对Map/List等集合支持就是全变成最普通的Hashmap或者ArrayList，一些指定的类型会丢失（例如LinkedHashMap-->HashMap），但是支持一些匿名的Map/List等集合类；
   而msgpack会保留集合类的类型（例如LinkedHashMap），但是不支持一些匿名集合类（例如List.subList(), Map.keySet()，Collections.emptyList()，Guava的匿名集合类，数据库查询结果直接返回的list）

当然JSF也是支持hessian协议的，具体参见 JSF客户端用户手册#序列化方式

如果需要支持字段顺序不一样的情况下调用：请使用hessian序列化，但为了性能及跨语言兼容性，请在保证 客户端与服务端的接口类文件保持一致的情况下使用msgpack序列化！！

接口类作为接口契约的重要组成部分，请尽量保证客户端与服务端的接口类文件的完全一致性（请引用同样的**jar**包！！），如果在bean中要增加新字段，请务必保证新字段加在字段序的最后！

1.2 区别之二：配置

为了消除歧义，标签和部分属性变化：

1. saf:protocol 换成了 jsf:server
2. saf:service 换成了 jsf:provider
3. saf:reference 换成了 jsf:consumer
4. 原来的group+version 合二为一换成了 alias

可通过工具转换，具体参见：JSF配置参考手册　JSF十分钟入门指南

1.3 区别之三 ： 关于方法重载

SAF支持方法重载，JSF因为支持跨语言调用，以及监控收集性能数据等原因，所以不支持重载；

参见关于SAF升级JSF中方法重载的说明

JSF FAQ#15.启动时提示Methodwithsamename"xxx"exists!Theusageofoverloadingmethodisdeprecated.

2. 服务端先升级到JSF

建议步骤如下：

1. 服务端无需修改业务代码，只需要引入jsf到pom里，然后添加一份jsf的spring配置； 参见JSF十分钟入门指南#3.2发布服务
2. 启动后既发布了SAF的Provider，也发布了JSF的Provider；
3. 通知客户端切到JSF，客户端无需修改业务代码，只需要引入jsf.pom，用同一个spring beanId的配置即可。 参见JSF十分钟入门指南#3.3调用服务
   此时saf客户端调saf，jsf客户端调jsf，服务端ref到同一个业务实现类即可。
4. 等到Consumer切换完成，删除SAF协议配置即可。

3. 客户端先升级到JSF

过程如下：

1. 数据同步worker自动将注册到SAF的zookeeper注册中心的Provider同步到JSF注册中心。（10分钟一次）
2. 此时服务端是SAF接口，调用端是JSF，修改配置如下：

原来的**saf**配置，**saf-consumer.xml**

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:saf="http://code.360buy.com/schema/saf"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
   http://code.360buy.com/schema/saf http://code.360buy.com/schema/saf/saf.xsd">
    <saf:registry id="jdZooKeeper" protocol="jdZooKeeper"
address="192.168.150.119:2181,192.168.150.120:2181,192.168.150.121:2181"/>
    <saf:reference id="helloService" interface="com.jd.testjsf.HelloService" group="ZG" version="0.0.1"
protocol="dubbo" />
</beans>
```

换成JSF的配置

jsf-consumer-dubbo.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:jsf="http://jsf.jd.com/schema/jsf"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
    http://jsf.jd.com/schema/jsf http://jsf.jd.com/schema/jsf/jsf.xsd">
    <!-- 连jsf的注册中心（已同步saf的provider） -->
    <jsf:registry id="jsfRegistry" index="i.jsf.jd.com" />

    <!--
        1. serialization配置为hessian或者java
        2. protocol配置为dubbo
        3. alias配置为 group:version （英文冒号隔开saf的group和version），如果服务端没有group，alias直接就是
version
    -->
    <jsf:consumer id="helloService" interface="com.jd.testjsf.HelloService" serialization="hessian"
alias="ZG:0.0.1" protocol="dubbo" />
</beans>
```

3. 业务代码无需更改，例如原来注入的接口代理类是 <saf:reference id="helloService" /> ，现在是 <jsf:consumer id="helloService" />

4. 等服务端切到JSF，再修改为走jsf协议而不是dubbo协议

其它参见 JSF与SAF的兼容与比较#JSF调用SAF（dubbo）

无