ConsumerGroup分组调用说明

工具

被章耿添加，被章耿最后更新于十月 20, 2016

自jsf-1.6.0开始，增加一种分组调用设置。对应spring标签为：<jsf:consumerGroup/> 对应API为com.jd.jsf.gd.config.ConsumerGroupConfig

实现原理：

其配置与 jsf:consumerGroup 除了alias可以配置多个外，其它配置与consumer一样。 增加一个 dstParam 参数。 参见JSF配置参考手册-<jsf:consumerGroup>

另外consumerGroup 下可以配置 consumer标签，用于覆盖分组属性；
配置 method 标签，用于覆盖方法属性；
配置 paramter 标签，用于覆盖自定义参数属性。

参考应用场景一：按机房调用，通过请求参数

例如：一个接口按机房分组，例如A机房一个组，B机房一个组，需要根据请求里的一个参数（机房），来调用到不同的分组。

此时接口里的某一个参数决定调那个分组，称之为路由参数。

第一步：首先，spring里的典型配置。

spring-consumer-group.xml

```
<!-- 一次性配置 4个分组，默认路由参数位置是第一个参数，默认超时是1000 -->
<jsf:consumerGroup id="helloService" interface="com.jd.testjsf.HelloService" alias="YZ,B28,YF,HC"
                   protocol="jsf" dstParam="0" timeout="1000">
    <jsf:consumer alias="HC" timeout="3000"/>  <!-- 假如这个分组所在机房较远，超时时间加大 -->
    <jsf:method name="echoStr" timeout="8888" dstParam="1"/> <!-- 假如这个方法的路由参数位置是第二个 -->
    <jsf:parameter key="token" value="I-am-token" hide="true"/> <!-- 假如这个接口有token -->
</jsf:consumerGroup>
```

当然在 consumerGroup可以为每个alias单独配置属性，甚至可以为每个alias下的接口下方法做单独配置。

如果需要alias自动适配，而不是直接写死，可以增加属性 <jsf:consumerGroup aliasAdaptive="true" /> 代表没找到要调用的分组时，自动增加引用（但是不会减少）。

第二步：在管理端配置 路由参数到分组的 映射规则。

服务管理--接口点进去--路由--新增路由

第三步：代码里使用

```
@Autowired
HelloService helloService;

或者
HelloService helloService= (HelloService) appContext.getBean("helloService");

helloService.sayHello("yz","xxxxx"); //第一参数是路由参数
helloService.echoStr("yyyyy","yz"); //第二参数是路由参数
```

参考应用场景二：按机房调用，通过隐式传参

例如：一个接口按机房分组，例如A机房一个组，B机房一个组，需要根据请求里的一个参数（机房），来调用到不同的分组。

此时接口没有参数决定调那个分组，新增参数改动太大。

第一步：首先，spring里的典型配置。

spring-consumer-group.xml

```
<!-- 一次性配置 4个分组，默认路由参数位置是第一个参数，默认超时是1000 -->
<jsf:consumerGroup id="helloService" interface="com.jd.testjsf.HelloService" alias="YZ,B28,YF,HC"
                   protocol="jsf" timeout="1000">
    <jsf:consumer alias="HC" timeout="3000"/>  <!-- 假如这个分组所在机房较远，超时时间加大 -->
    <jsf:parameter key="token" value="I-am-token" hide="true"/> <!-- 假如这个接口有token -->
</jsf:consumerGroup>
```

当然在 consumerGroup可以为每个alias单独配置属性，甚至可以为每个alias下的接口下方法做单独配置。

第二步：在管理端配置 路由参数到分组的 映射规则。

服务管理--接口点进去--路由--新增路由

第三步：代码里使用

```
@Autowired
HelloService helloService;
或者
HelloService helloService= (HelloService) appContext.getBean("helloService");

RpcContext.getContext().setAttachment(Constants.HIDDEN_KEY_DST_PARAM, "b28");//指定下次请求发到"b28"参数对应的分组
helloService.echoStr("xxxxx");

RpcContext.getContext().setAttachment(Constants.HIDDEN_KEY_DST_PARAM, "hc");//指定下次请求发到参数"hc"对应的分组
helloService.echoStr("yyyyy");
```

自定义路由不建议使用，请使用前两种方式来选择分组；

~~参考应用场景三：自定义路由~~

~~例如：分组目前是三个，但是未来可以是四个五个不一定，另外可能路由规则比较灵活，不再是简单的 参数A对应分组，有较为复杂的运算关系。~~

~~第一步：编写自定义路由规则，实现接口~~

```
package com.jd.testjsf.consumerGroup;

import com.jd.jsf.gd.client.GroupRouter;
import com.jd.jsf.gd.config.ConsumerConfig;
import com.jd.jsf.gd.config.ConsumerGroupConfig;
import com.jd.jsf.gd.msg.Invocation;
import com.jd.testjsf.HelloService;

import java.util.List;

public class TestGroupRouter implements GroupRouter {
    // 也可以继承 extends com.jd.jsf.gd.client.DefaultGroupRouter

    @Override
    public String router(Invocation invocation, ConsumerGroupConfig config) {
        List<String> aliases = config.currentAliases(); // 当前已有分组列表

        String clazz = invocation.getClazzName();  // 接口名
        String methodName = invocation.getMethodName(); // 方法名
        // 根据接口+方法名 可以拿到参数 是否有注解啥的。

        Object[] objects = invocation.getArgs(); // 也可以拿到调用参数，根据参数进行选择分组

        // 如果需要动态加分组
        config.addConsumerConfig("xxx1"); // 可以增加一个普通配置的分组引用
        ConsumerConfig newConfig = new ConsumerConfig();
        newConfig.setAlias("xxx2");
        newConfig.setTimeout(8888);
        config.addConsumerConfig(newConfig); // 可以增加一个自定义配置的分组引用
        // 如果需要动态减分组
        config.addConsumerConfig("yyy");
        // 也可以删除一个分组引用
        //config.delConsumerConfig("yyy");

        if (aliases.size() > 1) {
            if (HelloService.class.getName().equals(clazz)
                    && "echoStr".equals(methodName)) { // 例如这个方法
                String arg = (String) objects[0];
                String last = arg.substring(arg.length() - 1);
                try {
                    return aliases.get(Integer.parseInt(last) % 2);
                } catch (Exception e) {
                }
            }
        }
        return aliases.get(0);
    }
}
```

~~在com.jd.jsf.gd.client.GroupRouter#router方法中，可以拿到接口名，方法名，可以自行返回要调用的分组。~~

~~第三步：spring里的典型配置。~~

spring-consumer-group.xml

```
<!--自定义分组路由-->
<bean id="testGroupRouter" class="com.jd.testjsf.consumerGroup.TestGroupRouter"/>
<!--一次性配置4个分组，规则自定义的，默认超时是1000-->
<jsf:consumerGroup id="helloService"
                   interface="com.jd.testjsf.HelloService"
                   alias="YZ,B28,YF,HC"
                   protocol="jsf"
                   timeout="1000"
                   groupRouter="testGroupRouter">
    <jsf:consumer alias="HC" timeout="3000"/><!--假如这个分组所在机房较远，超时时间加大-->
    <jsf:parameter key="token" value="I-am-token" hide="true"/><!--假如这个接口有token-->
</jsf:consumerGroup>
```

~~第三步：代码里使用~~

```
@Autowired
HelloService helloService;

或者
HelloService helloService= (HelloService) appContext.getBean("helloService");

helloService.echoStr("xxxxx"); // 调用代码，会走入自定义router
```

无