API方式使用说明

工具

被章耿添加，被章耿最后更新于六月 09, 2015

使用说明

目前的API方式和Spring方式里的属性都是一一对应的。spring的方式无非就是spring转换为api的方式进行发布。

API方式类	spring标签
RegistryConfig	jsf:registry
ServerConfig	jsf:server
ProviderConfig	jsf:provider
ConsumerConfig	jsf:consumer
MethodConfig	jsf:method
ParameterConfig	jsf:parameter

参考代码

demo下载地址 JSF入门指南#2.DEMO下载

服务端

ServerMainAPI

```
package com.jd.testjsf;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import com.jd.jsf.gd.config.ProviderConfig;
import com.jd.jsf.gd.config.RegistryConfig;
import com.jd.jsf.gd.config.ServerConfig;

public class ServerMainAPI {

    private final static Logger LOGGER = LoggerFactory.getLogger(ServerMainAPI.class);

    public static void main(String[] args) {
        HelloService helloService = new HelloServiceImpl();
        // 注册中心实现（必须）
        RegistryConfig jsfRegistry = new RegistryConfig();
        jsfRegistry.setIndex("i.jsf.jd.com"); // 测试环境192.168.209.74 i.jsf.jd.com
        // jsfRegistry.setProtocol("jsfRegistry");
        // jsfRegistry.setAddress("192.168.209.74:40660");
        LOGGER.info("实例RegistryConfig");
        // 服务端配置（必须）
        ServerConfig serverConfig = new ServerConfig();
        serverConfig.setProtocol("jsf");
        // serverConfig.setPort(20880); // 可以指定端口
        LOGGER.info("实例ServerConfig");
        // 服务提供者属性
        ProviderConfig<HelloService> providerConfig = new ProviderConfig<HelloService>();
        providerConfig.setInterfaceId("com.jd.testjsf.HelloService");
        providerConfig.setAlias("CHANGE-IT");
        providerConfig.setRef(helloService);
        providerConfig.setServer(serverConfig); // 多个server用list
        // 如果外面已经有xml配的注册中心，使用providerConfig.setRegistry(RegistryFactory.getRegistryConfigs());
        providerConfig.setRegistry(jsfRegistry); // 多个registry用list,
        // providerConfig.setRegister(false);//打开注释表示不走注册中心
        LOGGER.info("实例ProviderConfig");
        // 暴露及注册服务
        providerConfig.export();
        LOGGER.info("服务端发布服务完成！");
        // providerConfig.unexport();
        // LOGGER.info("取消发布服务完成");
        // 启动本地服务，然后hold住本地服务
        synchronized (ServerMainAPI.class) {
            while (true) {
                try {
                    ServerMainAPI.class.wait();
                } catch (InterruptedException e) {
                }
            }
        }
    }
}
```

调用端

ClientMainAPI

```
package com.jd.testjsf;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import com.jd.jsf.gd.config.ConsumerConfig;
import com.jd.jsf.gd.config.RegistryConfig;

public class ClientMainAPI {

    private final static Logger LOGGER = LoggerFactory.getLogger(ClientMainAPI.class);

    public static void main(String[] args) {
        RegistryConfig jsfRegistry = new RegistryConfig();
        jsfRegistry.setIndex("i.jsf.jd.com"); // 测试环境192.168.209.74 i.jsf.jd.com
        // jsfRegistry.setProtocol("jsfRegistry");
        // jsfRegistry.setAddress("192.168.209.74:40660");
        LOGGER.info("实例RegistryConfig");
        // 服务提供者连接注册中心，设置属性
        ConsumerConfig<HelloService> consumerConfig = new ConsumerConfig<HelloService>();
        consumerConfig.setInterfaceId("com.jd.testjsf.HelloService");
        consumerConfig.setAlias("CHANGE-IT");
        consumerConfig.setProtocol("jsf");
        // 如果外面已经有xml配的注册中心，使用consumerConfig.setRegistry(RegistryFactory.getRegistryConfigs());
        consumerConfig.setRegistry(jsfRegistry);
        // consumerConfig.setUrl("jsf://127.0.0.1:20880;jsf://127.0.0.1:20881"); 直连
        LOGGER.info("实例ConsumerConfig");
        HelloService service = consumerConfig.refer();
        LOGGER.info("得到调用端代理：{}", service);

        while (true) {
            try {
                String result = service.echoStr("zhanggeng put");
                LOGGER.info("response msg from server :{}", result);
            } catch (Exception e) {
                LOGGER.error("", e);
            }
            try {
                Thread.sleep(2000);
            } catch (Exception e) {
            }
        }
    }
}
```

无