用户手册

张雄彪

Version 1.0, 2016-02-21

Table of Contents

功能简述

本手册简述了如何使用该jar包完成wsdl的解析及将第三方wsdl发布到dubbo中。**使用方需要直接依赖该jar包。**

依赖*jar*包描述

```
<!--axis1.4-->
<dependency>
  <groupId>org.apache.axis</groupId>
  <artifactId>axis</artifactId>
  <version>1.4</version>
</dependency>
<dependency>
  <groupId>org.apache.axis</groupId>
  <artifactId>axis-jaxrpc</artifactId>
  <version>1.4</version>
</dependency>

<dependency>
  <groupId>commons-discovery</groupId>
  <artifactId>commons-discovery</artifactId>
  <version>0.5</version>
</dependency>

<!-- dubbo -->
<dependency>
    <groupId>com.alibaba</groupId>
    <artifactId>dubbo</artifactId>
    <version>2.5.3</version>
    <exclusions>
        <exclusion>
            <groupId>org.springframework</groupId>
            <artifactId>spring</artifactId>
        </exclusion>
    </exclusions>
</dependency>

<!-- zookeeper client -->
<dependency>
  <groupId>com.github.sgroschupf</groupId>
  <artifactId>zkclient</artifactId>
  <version>0.1</version>
</dependency>

<!-- cxf -->
<dependency>
    <groupId>org.apache.cxf</groupId>
    <artifactId>cxf-rt-frontend-simple</artifactId>
```

```
    <version>2.6.1</version>
</dependency>
<dependency>
    <groupId>org.apache.cxf</groupId>
    <artifactId>cxf-rt-transports-http</artifactId>
    <version>2.6.1</version>
</dependency>
<dependency>
    <groupId>org.apache.cxf</groupId>
    <artifactId>cxf-rt-frontend-jaxws</artifactId>
    <version>2.6.1</version>
</dependency>
```

调用方*不需要*再在pom.xml中去描述这些依赖，直接依赖本jar对应的artifact即可。

Chapter 1. 注册第三方ws服务

功能描述

第三方ws服务，通常只有一个wsdl地址。此时可以通过本功能将ws服务发布到dubbo中。

实际本功能的操作是在中间产生一个代理，然后再发布成ws服务供消费方进行调用。

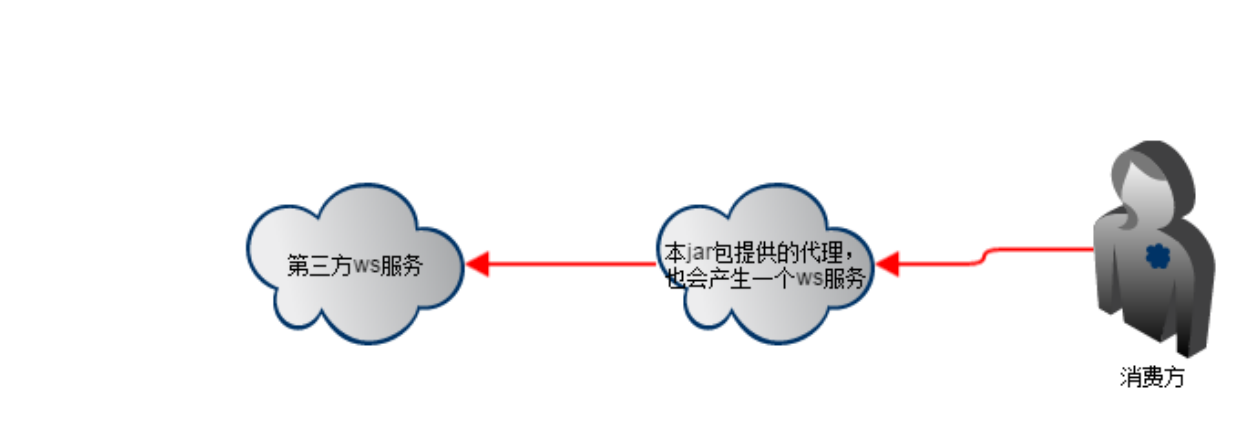

图1-调用示例

1.1. 使用spring配置文件加载注册Bean

在现有的jar包中已经将已经配置好的配置文件打包了，详见jar包根目录。

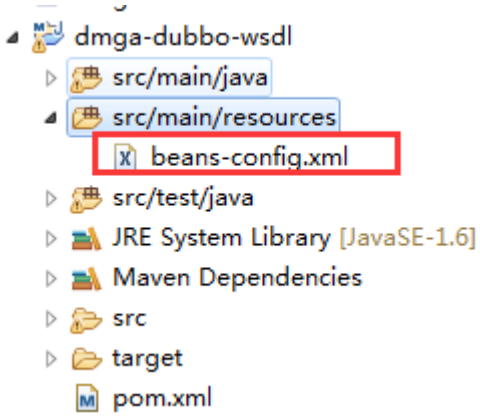

图2-工程中配置文件路径

配置文件内容简述

```
<bean id="defaultEnv" class="com.dm.dubbo.wsdl.registry.domain.Environment">
  <property name="registryAddress" value="zookeeper://localhost:1234"></property> ③
  <property name="protocolName" value="webservice"></property>
  <property name="protocolPort" value="8081"></property>
  <property name="protocolServer" value="servlet"></property>
  <property name="appName" value="WsGenericService"></property>
</bean>

<bean id="registryService"
  class="com.dm.dubbo.wsdl.registry.service.impl.RegistryServiceImpl"> ①
  <property name="env" ref="defaultEnv" /> ②
</bean>
```

① 注册服务的核心Bean，调用方直接使用该Bean的方法即可。

② 注册服务时使用的默认配置环境。

③ ZooKeeper的地址配置

上述是一个示例配置，配置文件已经在jar包中。**调用方调用时需要在web.xml中手动配置加载该配置文件**，否则将不会生效。

1.2. 调用注册Bean完成注册

如上所述，当注册服务Bean配置好了后，调用方，只需要直接调用对应的registry方法即可。

示例代码

```
@Autowired
private RegistryService registryService;

@Test
public void testRun(){
    WsEnvironment context = new WsEnvironment();
    context.setId(UUID.randomUUID().toString());
    context.setGroup("ServerID"); ①
    context.setVersion("1.0.0"); ②
    context.setWsdlAddress(
"http://localhost:8080/eview/services/GabTerminalDataServer?wsdl"); ③
    context.setTargetNamespace("http://server.webservice.eview.dm.com"); ④
    context.setClientClassName(
"com.dm.dubbo.wsdl.registry.service.impl.GenericServiceAxis1Impl"); ⑤
    context.setAppName("GabTerminalDataServer");
    context.setProtocolPort(8081);
    try {
        @SuppressWarnings("rawtypes")
        ServiceConfig serviceConfig = registryService.registry(context); ⑥
        assertNotNull(serviceConfig);
        System.in.read();
    } catch (Exception e) {
        e.printStackTrace();
        fail();
    }
}
```

① 设置服务方ID

② 设置服务方版本号

③ 设置第三方的wsdl地址

④ 设置第三方wsdl描述中的命名空间（此处可不设置）

⑤ 设置调用第三方ws服务时使用的客户端。此处客户端类名必须与对应的ws服务使用的框架版本一致

⑥ 完成服务的发布及注册

1.3. web工程中添加cxf的拦截

web.xml

```
<servlet>
        <servlet-name>dubbo</servlet-name>
        <servlet-class>
com.alibaba.dubbo.remoting.http.servlet.DispatcherServlet</servlet-class>
        <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
        <servlet-name>dubbo</servlet-name>
        <url-pattern>/*</url-pattern>
</servlet-mapping>
```

1.4. 日志记录

由于需要集成在调用ws服务时完成日志的记录，所以需要在本jar的调用端完成日志记录写库。

如果在本jar中将直接与数据库打交道将不利于本jar的移植性。**所以日志记录功能将交给调用方来实现。**

示例

```
package com.dm.dubbo.wsdl.registry.service.impl;

import java.util.Map;

import com.dm.dubbo.wsdl.registry.intercept.service.PreProcessService; ①

public class LogServiceImpl implements PreProcessService {

    @Override
    public Object process(Map<String, Object> params) throws Exception { ②
        System.out.println("clientIp:" + params.get("clientIp"));
        System.out.println("methodName:" + params.get("methodName"));
        System.out.println("condition:" + params.get("condition"));
        return null;
    }

}
```

① 调用方需要实现PreProcessService

② 每一次接口调用时，都会将参数传入到params中，调用会取出后写入库即可。

调用方实现了该接口后，需要将该类声明为springBean，使用*注解或XML配置*。 例如：

```
<bean class="com.dm.dubbo.wsdl.registry.service.impl.LogServiceImpl"
></bean>
```

Chapter 2. 解析ws服务结构

在注册第三方ws服务过程中，通常对方只会提供一个wsdl地址。此时就需要手动解析该ws服务中有哪些方法及对应的参数列表。

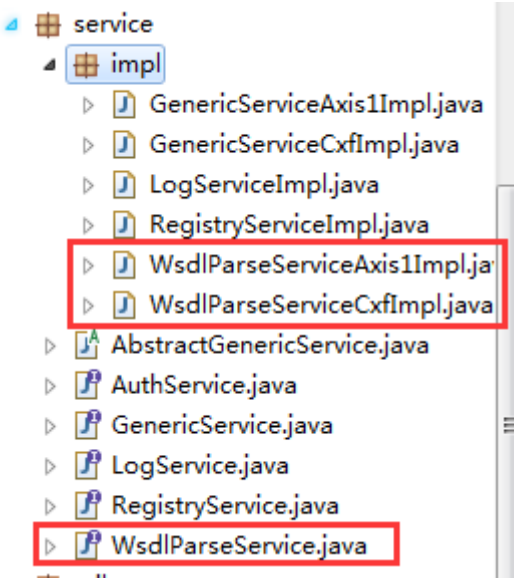

图3-类文件

2.1. 解析axis1.4版本的ws文档

解析axis1.4时，只需要调用对应的实现类即可。该实现类建议使用spring配置并注入。

调用示例

```
package com.dm.dubbo.wsdl.registry.service.impl;

import static org.junit.Assert.*;

import org.junit.Test;

import com.dm.dubbo.wsdl.registry.domain.WsService;

public class WsdlParseServiceAxis1ImplTest {

    @Test
    public void testParse(){
        WsdlParseServiceAxis1Impl axis = new WsdlParseServiceAxis1Impl();
        try {
            WsService wsService = axis.parse(
"http://localhost:8080/eview/services/GabTerminalDataServer?wsdl");
            wsService.print();
        } catch (Exception e) {
            e.printStackTrace();
            fail();
        }
    }
}
```

2.2. 解析axis2及cxf版本的ws文档

调用示例

```
package com.dm.dubbo.wsdl.registry.service.impl;

import static org.junit.Assert.*;

import org.junit.Test;

import com.dm.dubbo.wsdl.registry.domain.WsService;

public class WsdlParseServiceCxfImplTest {

    @Test
    public void testParse(){
        WsdlParseServiceCxfImpl cxf = new WsdlParseServiceCxfImpl();
        try {
            WsService wsService = cxf.parse(
"http://localhost:8080/dsp/services/TerminalDataServer?wsdl");
            wsService.print();
        } catch (Exception e) {
            e.printStackTrace();
            fail();
        }
    }
}
```

解析完成后将会得到一个`WsService`对象。

WsService 描述服务信息

```
/** 命名空间 */
private String targetNameSpace;

/** wsdl地址 */
private String wsdlUrl;

/** 服务名称 */
private String serviceName;

/** 方法集合 */
private List<WsOperation> opList; ①
```

① 包含方法集合

WsOperation 方法信息

```
/** 方法名称 */
private String name;

/** 输入参数 */
private List<WsArgument> inputArgs; ①

/** 返回类型 */
private String returnType;
```

① 包含参数列表

WsArgument 参数信息

```
private String name; // 参数名称

private String type; // 参数类型
```