# 根据wsdl生成java代码

## 问题描述

在调用外围系统的webservice服务的时候，一般来说对方会提供wsdl文件，然后我们要根据wsdl，然后我们要根据wsdl生成相应的java类，通过这种方式来做到跨语言的调用。

在java中，我们可以使用wsimport工具或者其它第三方工具如wsdl4j工具类生成。

使用jdk里面自带的wsimport工具，具体使用方法如下：

在jdk的bin目录下有wsimport.exe,使用如下命令：

wsimport -s F:\temp -p com.client.import http://localhost:6666/service/interpret?wsdl

常用的参数有：

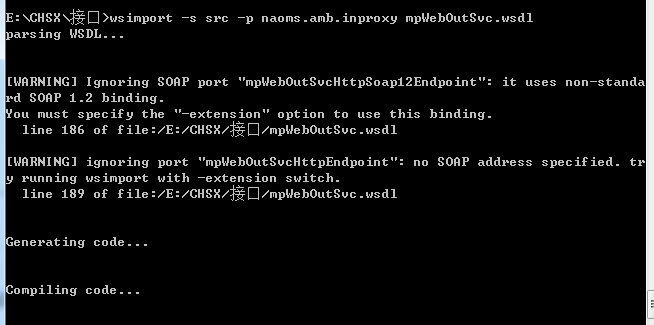
-d <directory> 在指定的目录中生成class文件

-clientjar <jarfile> 在当前目录生成jar文件，结合-d参数可以在指定的目录生成jar文件

-s <directory> 在指定的目录中生成java源文件

-p <pkg> 指定生成文件的包结构

-keep 在生成class文件或者jar包的同时，保留java源文件



## CXF

Wsimport.exe只能解析soap协议1.1，不能解析soap1.2协议

建议使用cxf。CXF提供的wsdl2java.exe能够解析soap协议的1.1和1.2版本

CXF可以构建服务的前端api。支持 a variety of protocols such as SOAP, XML/HTTP, RESTful HTTP, or CORBA and work over a variety of transports such as HTTP, JMS or JBI.

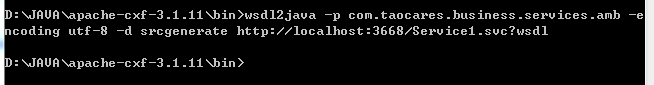
官方网址：<http://cxf.apache.org/>

文档地址: http://cxf.apache.org/docs/index.html



使用方法

wsdl2java –p 包名 -encoding utf-8 –d 生成代码文件输出目录 wsdlurl



在使用wsdl2java 命令生成java 客户端或者服务器端java 对象时，有时候会报异常：具有相同名称 "xxx" 的类/接口已在使用。请使用类定制设置来解决此冲突。 这是由于 wsdl 文档中有重复的元素导致的。解决此问题只需要在wsdl2java 命令中新增参数： -autoNameResolution 即可， 这样就可以解决类名冲突问题。

wsdl2java -p ws.client -d d:/Java -autoNameResolution <http://xxx.xxx.xxx.xxx:8080/xxxService?wsdl>

客户端调用

AirlineReceiveServiceImplService airlineReceiveServiceImplService = **new** AirlineReceiveServiceImplService(**new** URL("http://192.168.163.241:8080/naoms.server/fimsService/airline?wsdl"));

IAirlineReceiveService airlineReceiveService = airlineReceiveServiceImplService.getAirlineReceiveServiceImplPort();

//调用方法 airlineReceiveService.addFlightByAirline(flight, shareDtos, routeDtos, airlineData)

### 添加拦截器

Pom中添加maven依赖

<!-- cxf支持 -->

<dependency> <groupId>org.apache.cxf</groupId> <artifactId>cxf-rt-frontend-jaxws</artifactId> <version>3.1.10</version> </dependency>

<dependency> <groupId>org.apache.cxf</groupId> <artifactId>cxf-rt-transports-http</artifactId> <version>3.1.10</version> </dependency>

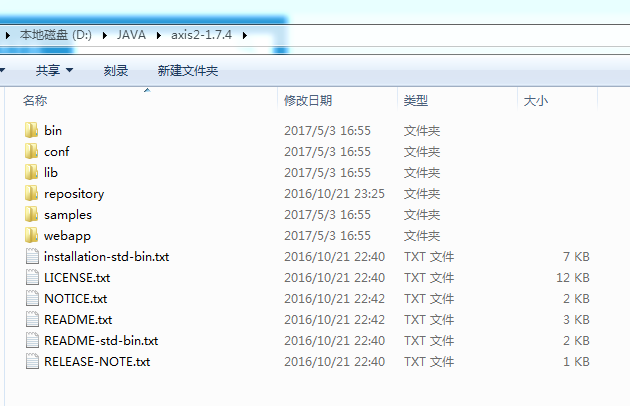




## AXIS

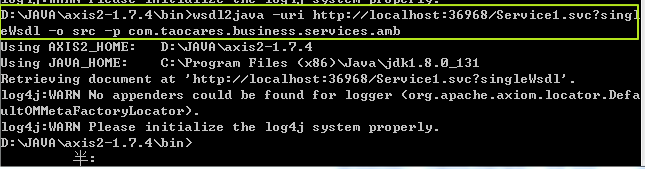
网站：http://ws.apache.org/axis2/

Axis is essentially a SOAP engine -- a framework for constructing SOAP processors such as clients, servers, gateways, etc. The current version of Axis is written in Java, but a C++ implementation of the client side of Axis is being developed.



wsdl2java -uri http://localhost:36968/Service1.svc?singleWsdl -o src -p com.taocares.business.services.amb

-o 指定输出目录 -p 指定包



客户端调用在pom.xml中添加

<!-- axis2 支持 -->

<dependency>

<groupId>org.apache.axis2</groupId>

<artifactId>axis2</artifactId>

<version>1.7.4</version>

<type>pom</type>

<scope>compile</scope>

</dependency>

<dependency>

<groupId>org.apache.axis2</groupId>

<artifactId>axis2-adb</artifactId>

<version>1.7.4</version>

<type>jar</type>

<scope>compile</scope>

</dependency>

<dependency>

<groupId>org.apache.axis2</groupId>

<artifactId>axis2-kernel</artifactId>

<version>1.7.4</version>

<type>jar</type>

<scope>compile</scope>

</dependency>

<dependency>

<groupId>org.apache.axis2</groupId>

<artifactId>axis2-transport-http</artifactId>

<version>1.7.4</version>

</dependency>

<dependency>

<groupId>org.apache.axis2</groupId>

<artifactId>axis2-transport-local</artifactId>

<version>1.7.4</version>

</dependency>

<!-- axis2 支持 -->



客户端调用

MpWebOutSvcStub webOutSvcStub = **new** MpWebOutSvcStub(PropertiesUtils.*getAmb\_url*());

Options options = webOutSvcStub.\_getServiceClient().getOptions();

//超时时间

options.setTimeOutInMilliSeconds(2000);

//session保持

options.setManageSession(**true**);

//禁用分段传输

options.setProperty(org.apache.axis2.transport.http.HTTPConstants.***CHUNKED***, **false**);

//调用amb的receiveMessage接口

ReceiveMessageResponse msgResponse = webOutSvcStub.receiveMessage();

MsgOutRespObj respObj = msgResponse.get\_return();