# 前提

什么是WebService

Web service是一个平台独立的，低耦合的，自包含的、基于可编程的web的应用程序，可使用开放的XML（标准通用标记语言下的一个子集）标准来描述、发布、发现、协调和配置这些应用程序，用于开发分布式的互操作的应用程序。

Web Service技术， 能使得运行在不同机器上的不同应用无须借助附加的、专门的第三方软件或硬件， 就可相互交换数据或集成。依据Web Service规范实施的应用之间， 无论它们所使用的语言、 平台或内部协议是什么， 都可以相互交换数据。Web Service是自描述、 自包含的可用网络模块， 可以执行具体的业务功能。Web Service也很容易部署， 因为它们基于一些常规的产业标准以及已有的一些技术，诸如标准通用标记语言下的子集XML、HTTP。Web Service减少了应用接口的花费。Web Service为整个企业甚至多个组织之间的业务流程的集成提供了一个通用机制。

调用网络上的WebService服务网址：<http://webxml.com.cn/>

# SOAP和WSDL概念

SOAP作为一个基于XML语言的协议用于在网上传输数据，是基于HTTP协议的

简单对象访问协议

SOAP的xml规范(约束)的组成如下：

Envelope – 必须的部分。以XML的根元素出现

Headers – 可选的

Body – 必须的。在body部分，包含要执行的服务器的方法。和发送到服务器的数据。

基于http协议

请求行

POST /WebServices/IpAddressSearchWebService.asmx HTTP/1.1

请求头

Host: ws.webxml.com.cn

Content-Type: text/xml; charset=utf-8

Content-Length: length

SOAPAction: "http://WebXml.com.cn/getCountryCityByIp"

以下是请求体

<?xml version="1.0" encoding="utf-8"?>

<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">

<soap:Body>

<getCountryCityByIp xmlns="http://WebXml.com.cn/">

<theIpAddress>string</theIpAddress>

</getCountryCityByIp>

</soap:Body>

</soap:Envelope>

WSDL 描述文档告诉你此请求的一些信息：由xml格式组成

WSDL(WebService Description Language):web 服务描述语言

就是一个xml文档，用于描述当前服务的一些信息（服务名称、服务的发布地址、服务提供的方法、方法的参数类型、方法的返回值类型等）

wsdl又可以叫：webService的使用说明书

# 基于jdk1.7发布一个WebService服务

## 服务端发布

步骤

第一步：创建一个Java项目

注意：这里不使用web项目，不用导第三方jar包

第二步：创建一个类，加入Webservice注解 ，添加方法

注意 要普通的方法：不能是静态方法

@javax.jws.WebService //类中使用此注解

public class A{

public String str(String name,int i) {

System.out.println("服务端被调用了"+i+"次");

return "您好！"+name+"欢迎您使用此weservice";

}

//main方法

public static void main(String[] args) {

// 参数1：必须带ip和端口、访问路径(可以随便定义一般都写类名)三样不可缺少

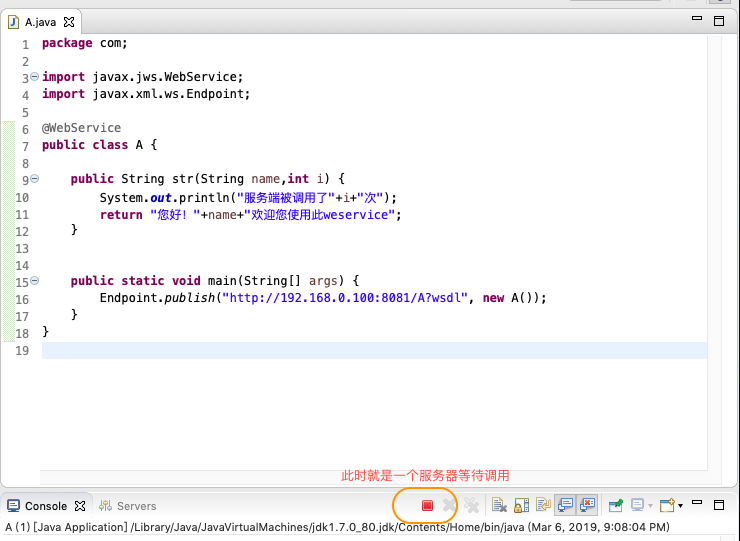
//参数2：类的对象

javax.xml.ws.Endpoint.publish("http://192.168.0.101:8081/A", new A()); //地址可以用来访问

}

}

第三步 直接运行主方法即可发布：此时主方法就在运行中(就是一个服务)等待调用

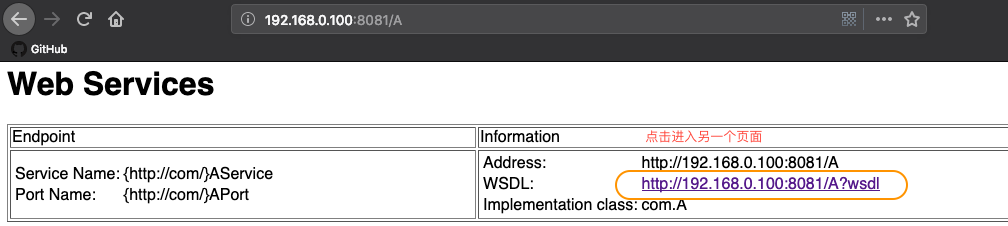


注意：不是调用一次就会停止的，永久服务

第四步 测试

发布服务后JDK就已经给你设置了WSDL(使用描述说明书)的访问了

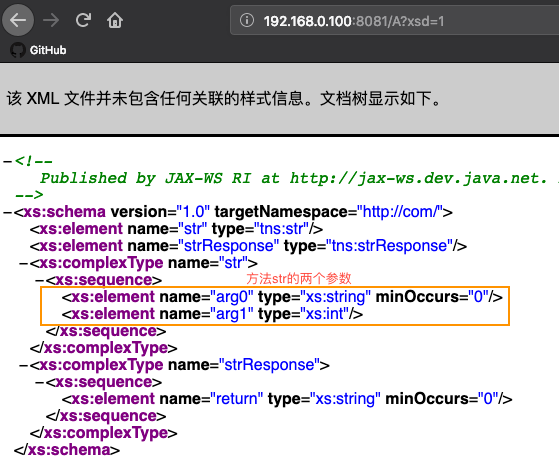
此时就可以通过浏览器url就可以进行访问了：<http://192.168.0.101:8081/A>



进入<http://192.168.0.101:8081/A>？wsdl



进入方法的详细页面：这里有点不同，貌似JDK发布的才有这个页面的



## 客户端调用

第一步 使用jdk7中bin目录下的命令wsimport解析生成代码

常用命令参数：

-s d:/wwl/test/ -s生成的文件(有class和源文件java)指定路径目录

-p wwl.service -p指定类的全路径，不使用此参数，那么生成的类的路径和服务端的一致

-d d:/wwl/test/ -d<指定目录生成class文件目录> 默认参数一般都不用写

注意 使用命令解析的时候服务端一定要开启

Wsimport –s . –p wwl.service http://192.168.0.2:8081/A?wsdl 注意-s 和-p之间的点都有空格

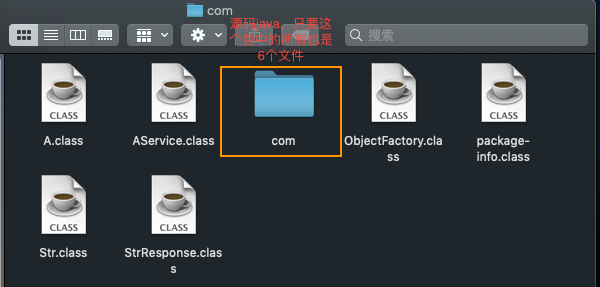
命令意思是：在当前路径(根目录)生成文件夹wwl/service/6个java和6个编译好的class(类中包路径：package wwl.service;)

注意：http://192.168.0.2:8081/A?wsdl 这个网址是在服务端自己定义的：服务端的ip和端口 + 工程名和默认参数wsdl

wsimport -s com http://192.168.0.100:8081/A?wsdl

命令意思是：在com的文件夹下生成文件，和服务端调用的类路径一致

注意：需要自己先创建文件夹，不然报错



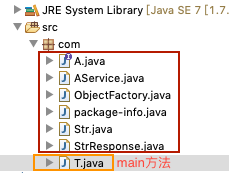
第二步 把生成的java类导入工程中的

大概有以下些类（了解）

A接口类 和服务端要调用的类名一致

AService ：调用需要用到类，继承Service

ObjectFactory，package-info ，Str（服务端方法名，这里变成类）



第三步 类中的使用

public static void main(String[] args) {

//第一 使用远程的类名+Service（默认会自动加一个Service）创建对象

AService a= new AService();

//第二 创建客户端代理对象，用于远程调用

A porxy = a.getAPort();

//第三 调用远程不是静态类的方法（静态类方法不能调用）

String str = porxy.str("你好", 1);

System.out.println(str);

}

注意 服务端不会就调用一次就停止的，永久服务

# CXF框架

## 前提

CXF 框架是一种基于 Servlet 技术的 SOA(面向服务) 应用开发框架

什么是CXF？

Apache CXF = Celtix + XFire，开始叫 Apache CeltiXfire，后来更名为 Apache CXF 了，以下简称为 CXF。CXF 继承了 Celtix 和 XFire 两大开源项目的精华，提供了对 JAX-WS 全面的支持，并且提供了多种 Binding 、DataBinding、Transport 以及各种 Format 的支持，并且可以根据实际项目的需要，采用代码优先（Code First）或者 WSDL 优先（WSDL First）来轻松地实现 Web Services 的发布和使用。Apache CXF已经是一个正式的Apache顶级项目。

Apache CXF 是一个开源的 Services 框架，CXF 帮助您利用 Frontend 编程 API 来构建和开发 Services ，像 JAX-WS 。这些 Services 可以支持多种协议，比如：SOAP、XML/HTTP、RESTful HTTP 或者 CORBA ，并且可以在多种传输协议上运行，比如：HTTP、JMS 或者 JBI，CXF 大大简化了 Services 的创建，同时它继承了 XFire 传统，一样可以天然地和 Spring 进行无缝集成。

下载jar包：此webSservice框架是apache开源的：官网：cxf.apache.org

版本问题：cxf3.2.6测试需要JDK1.8才行

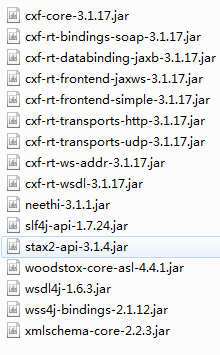
## 服务端发布

为啥要用CXF呢？因为JDK提供的不能用web工程，只能使用main发布

第一步 下载jar包，导入jar包

官网：cxf.apache.org经测试： 版本3.2以上的貌似都要jdk1.8

本次导包：需要spring 4.1.9的基本框架如：web等，CXF版本3.1.17所需jar如下：



maven工程

<dependencies>

<!-- spring框架的基本包 -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>4.3.13.RELEASE</version>

</dependency>

<!-- spring的web包 -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-web</artifactId>

<version>4.3.13.RELEASE</version>

</dependency>

<!-- CXF的 http包 -->

<dependency>

<groupId>org.apache.cxf</groupId>

<artifactId>cxf-rt-transports-http</artifactId>

<version>3.1.17</version>

</dependency>

<!-- CXF 的基本包 -->

<dependency>

<groupId>org.apache.cxf</groupId>

<artifactId>cxf-rt-frontend-jaxws</artifactId>

<version>3.1.17</version>

</dependency>

</dependencies>

第二步 WEB.xml 配置

配置servlet进行加载CXF框架

<servlet>

<servlet-name>cxf</servlet-name>

<servlet-class>org.apache.cxf.transport.servlet.CXFServlet</servlet-class>

<!-- 通过初始化参数指定CXF框架的配置文件位置 -->

<init-param>

<param-name>config-location</param-name>

<param-value>classpath:cxf.xml</param-value>

</init-param>

</servlet>

<servlet-mapping>

<servlet-name>cxf</servlet-name>

<url-pattern>/service/\*</url-pattern> 要发布服务的路径(访问此servlet的路径)

</servlet-mapping>

第三步 编写需要开发的接口类和实现类

必须接口类实现WebService注解

@WebService 注意此注解一定要在接口中写

public interface CxfService {

public String a(String name,int i);

}

实现类 和普通类没什么区别，方法中如果有不想发布的方法写成静态类或者百度

public class CxfServiceImpl implements CxfService{

public String a(String name, int i) { //发布的方法

System.out.println("使用cxf发布的服务被调用了"+i+"次");

return name;

}

}

第四步 cxf.xml的配置

<?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:jaxws="http://cxf.apache.org/jaxws" <!—CXF的约束文件 可以在核心jar中找到-->

xmlns:context="http://www.springframework.org/schema/context"

xsi:schemaLocation="http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans-3.0.xsd

http://cxf.apache.org/jaxws

http://cxf.apache.org/schemas/jaxws.xsd

http://www.springframework.org/schema/context

http://www.springframework.org/schema/context/spring-context.xsd">

<bean id="cxfService" class="wwl.CxfServiceImpl"/>

<!-- address属性值：‘http://ip端口/工程名/service/test?wsdl’中的test,这个test名字随便写 -->

<jaxws:server id="myService" address="/test"> 这个id在客户端中用的到

<jaxws:serviceBean>

<ref bean="cxfService"/>

</jaxws:serviceBean>

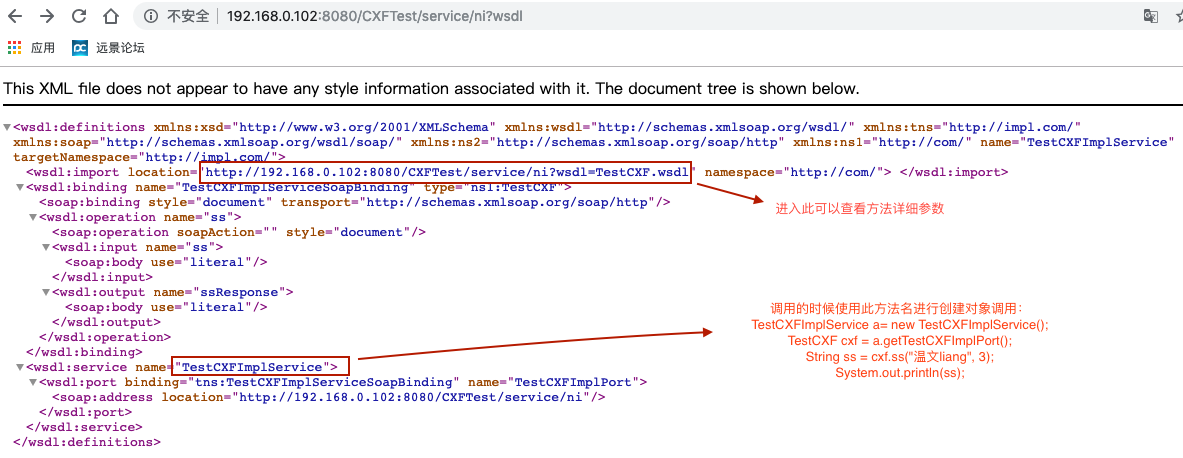
</jaxws:server>

</beans>

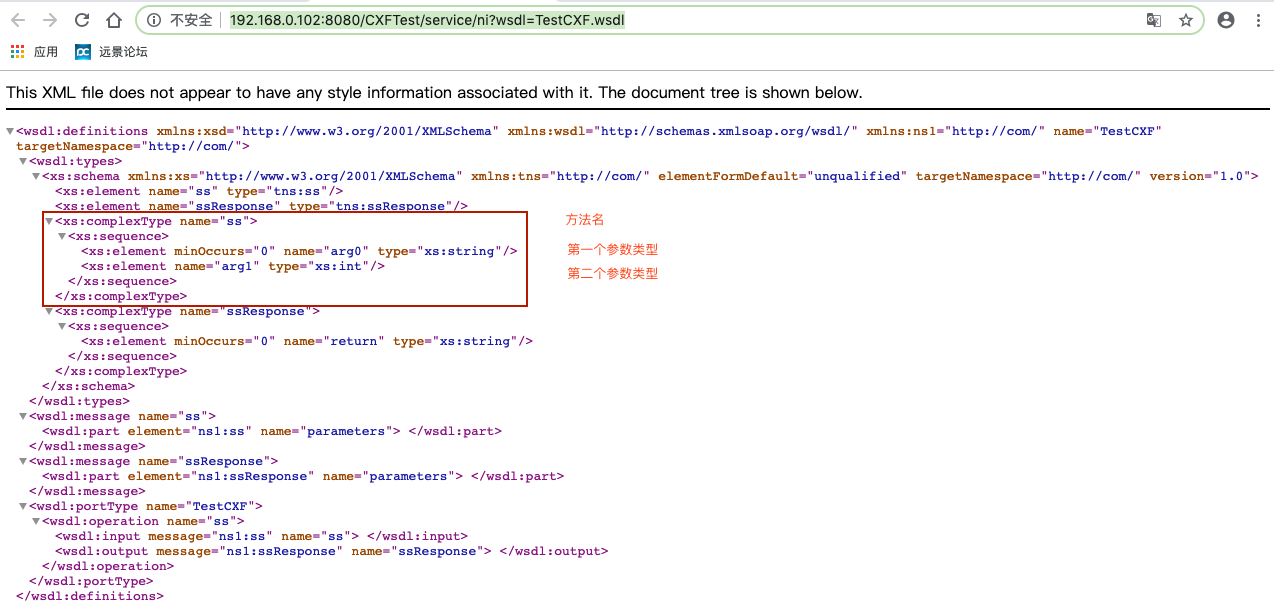
第五步 测试 和JDK发布wsdl(使用说明书)的有些不一样

访问：http://192.168.0.102:8080/CXFTest/service/ni此路径没用的

访问：http://192.168.0.102:8080/CXFService/service/test?wsdl 和JDK的不一样，和服务的一样



http://192.168.0.102:8080/CXFTest/service/ni?wsdl=TestCXF.wsdl页面



## 客户端调用

webservice设计就是跨平台垮语言设计的，所以调用方式都是通用的

第一种方式就是原始的webService进行调用（看上面JDK客户端调用方式）

第二种方式 使用CXF框架整合spring调用

第一步 和服务端一样 下载jar包，导入jar包

官网：cxf.apache.org经测试： 版本3.2以上的貌似都要jdk1.8

本次导包：需要spring 4.1.9的基本框架如：web等，CXF版本3.1.17所需jar如下：

maven工程

<dependencies>

<!-- spring框架的基本包 -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>4.3.13.RELEASE</version>

</dependency>

<!-- spring的web包 -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-web</artifactId>

<version>4.3.13.RELEASE</version>

</dependency>

<!-- CXF的 http包 -->

<dependency>

<groupId>org.apache.cxf</groupId>

<artifactId>cxf-rt-transports-http</artifactId>

<version>3.1.17</version>

</dependency>

<!-- CXF 的基本包 -->

<dependency>

<groupId>org.apache.cxf</groupId>

<artifactId>cxf-rt-frontend-jaxws</artifactId>

<version>3.1.17</version>

</dependency>

</dependencies>

第二步 WEB.xml 配置 和服务端配置一样

<servlet>

<servlet-name>cxf</servlet-name>

<servlet-class>org.apache.cxf.transport.servlet.CXFServlet</servlet-class>

<!-- 通过初始化参数指定CXF框架的配置文件位置 -->

<init-param>

<param-name>config-location</param-name>

<param-value>classpath:cxf.xml</param-value>

</init-param>

</servlet>

<servlet-mapping>

<servlet-name>cxf</servlet-name>

<url-pattern>/service/\*</url-pattern> 要发布服务的路径(访问此servlet的路径)

</servlet-mapping>

第三步 使用wsimport命令或者CXF提供wsdl2java生成本地代码，只需要生成的接口文件一个即可

常用命令参数：

-s d:/wwl/test/ -s<指定目录生成java文件目录>

-p wwl.service -p<生成类的包名>

-d d:/wwl/test/ -d<指定目录生成class文件目录> 默认参数一般都不用写

注意 使用命令解析的时候服务端一定要开启

获得一系列的类后：只要接口类 放入工程中



此时会报错，进入此接口删除(看图)



第四步 cxm.xml的配置

<?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:jaxws="http://cxf.apache.org/jaxws"

xmlns:context="http://www.springframework.org/schema/context"

xsi:schemaLocation="http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans-3.0.xsd

http://cxf.apache.org/jaxws

http://cxf.apache.org/schemas/jaxws.xsd

http://www.springframework.org/schema/context

http://www.springframework.org/schema/context/spring-context.xsd">

<jaxws:client id="demo"

address="http://192.168.0.102:8080/CXFService/service/test" address：服务的地址(没有参数?wsdl)

serviceClass="wwl.service.CxfService"> serviceClass:解析得到的接口类

</jaxws:client>

</beans>

第五步 类中调用操作

public class Test {

public static void main(String[] args) {

ApplicationContext context = new ClassPathXmlApplicationContext("cxf.xml");

CxfService bean = (CxfService) context.getBean("demo"); //调用cxm.xml中配置的id属性值

String str = bean.a("wenwenliang", 1); //服务端中发布的方法名

System.out.println(str);

}

}

# 网络服务上的调用

WEB服务网址

<http://www.webxml.com.cn/zh_cn/web_services.aspx>



后续都和普通调用一致的，请查看调用操作