# 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

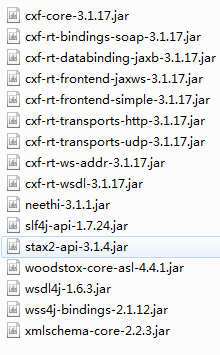
版本问题：cxf3.2.6测试需要JDK1.8才行

# 服务端发布

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

本次导入：CXF版本3.1.17 spring 4.1.9，所需jar如下：



第二步 WEB.xml 配置

需使用servlet进行加载CXF框架

小知识：可使用上下文来加载cxf.xml,不同之处：上下文加载配置文件加载：服务启动就加载进内存，在servlet配置路径加载，在第一次使用才加载此配置文件至内存

<context-param> 上下文加载配置文件

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

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

</context-param>

<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

</servlet-mapping>

第二步 编写需要开发的接口类和实现类

接口类

@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;

}

}

第三步 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>

第四步 测试 和原始webService一样

访问：http://192.168.0.102:8080/CXFService/service/test?wsdl 如果能出现wsdl页面证明成功发布

# 客户端调用

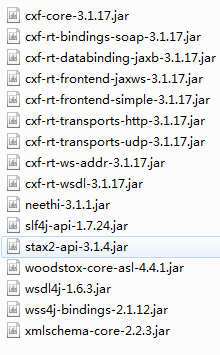
第一种方式就是原始的webService进行调用（见分布式中的webSevice学习文件）

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

本次导入：CXF版本3.1.17 spring 4.1.9，所需jar如下：



第二步 WEB.xml 配置 和服务端配置一样

需使用servlet进行加载CXF框架

小知识：可使用上下文来加载cxf.xml,不同之处：上下文加载配置文件加载：服务启动就加载进内存，在servlet配置路径加载，在第一次使用才加载此配置文件至内存

第三步 使用wsimport命令或者CXF提供wsdl2java生成本地代码，只需要生成接口文件

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

wsimport –s . –p wwl.service http://192.168.0.2:8081/MyWeServer?wsdl 命令意思是：在本目录生成包名为wwl.service的类

注意：http://192.168.0.2:8081/MyWeServer?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

获得一系列的类后：只要接口类 放入工程中

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">

<!-- 客户端调用 id属性用来调用 在类中需要使用此属性

address：服务的地址 没有参数?wsdl

serviceClass:解析得到的接口类-->

<jaxws:client id="demo"

address="http://192.168.0.102:8080/CXFService/service/test"

serviceClass="wwl.service.CxfService">

</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);

}

}