[WebService--CXF以及CXF与Spring的整合（jaxws:server形式配置）](https://www.cnblogs.com/zhanxiaoyun/p/6144651.html)

作者：[@展云](http://www.cnblogs.com/zhanxiaoyun/)  
本文为作者原创，转载请注明出处：<https://www.cnblogs.com/zhanxiaoyun/p/6144651.html>

**目录**

[**1.服务端**](https://www.cnblogs.com/zhanxiaoyun/p/6144651.html#_caption_0)  
[**2.客户端**](https://www.cnblogs.com/zhanxiaoyun/p/6144651.html#_caption_1)  
[**1.服务端**](https://www.cnblogs.com/zhanxiaoyun/p/6144651.html#_caption_2)  
[**2.客户端**](https://www.cnblogs.com/zhanxiaoyun/p/6144651.html#_caption_3)  
[**1.新建一个maven项目（写过一个随笔），并且加入jar包**](https://www.cnblogs.com/zhanxiaoyun/p/6144651.html#_caption_4)  
[**2.让spring管理ServerFactoryBean**](https://www.cnblogs.com/zhanxiaoyun/p/6144651.html#_caption_5)  
[**3.cxf集成到web容器中**](https://www.cnblogs.com/zhanxiaoyun/p/6144651.html#_caption_6)  
[**4.启动web容器发布webservice服务**](https://www.cnblogs.com/zhanxiaoyun/p/6144651.html#_caption_7)

前言：好记性不如烂笔头，写博客的好处是，以前接触的东西即便忘记了，也可以从这里查找。

　　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 。可以与Spring进行快速无缝的整合。灵活的部署，可以运行在Tomcat,Jboss,Jetty(内置),IBMWS,BeaWL上面。

　　更多CXF介绍：http://www.ibm.com/developerworks/cn/education/java/j-cxf/

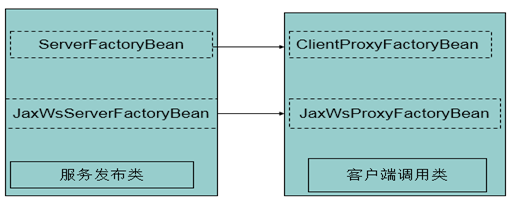
**一、cxf发布服务的类**

用两个不同的类发布应用：

　　a. ServerFactoryBean   -- FacotryBean

　　b. JaxWsServerFactoryBean(建议使用此类)

服务端与客户端类的使用映射关系如下图所示：



**二、使用ServerFactoryBean类发布标准的webservice服务**

　　下载cxf安装包apache-cxf-2.4.2，里面lib下jar包以供项目所有。

**1.服务端**

a.新建web项目，加入cxf的jar包

b.定义webservice的接口

package com.wp.service;

import javax.jws.WebService;

@WebService

public interface HelloWs {

public String sayHello(String name);

}

c.定义webservice接口的实现类

[复制代码](javascript:void(0);)

package com.wp.service;

public class HelloWsImpl implements HelloWs {

@Override

public String sayHello(String name) {

return "hello" + name;

}

}

[复制代码](javascript:void(0);)

d.发布webservice服务

[复制代码](javascript:void(0);)

package com.wp.service;

import org.apache.cxf.frontend.ServerFactoryBean;

public class Server {

public static void main(String[] args) {

ServerFactoryBean sfb = new ServerFactoryBean();

//1.服务提供者实现的接口

sfb.setServiceClass(HelloWs.class);

//2.指定访问路径

sfb.setAddress("http://localhost:9090/ws");

//3.指定服务实现类

sfb.setServiceBean(new HelloWsImpl());

//4.发布

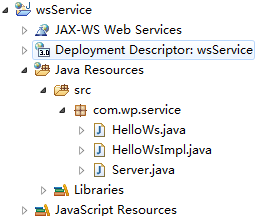
sfb.create();

System.out.println("发布成功...");

}

}

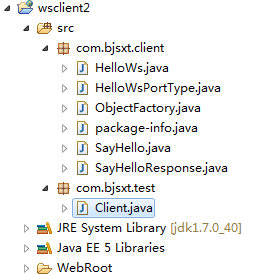
[复制代码](javascript:void(0);)



**2.客户端**

a.使用wsdl2java生成客户端代码调用服务（之前有解释，这里就不做解释了）

https://images2015.cnblogs.com/blog/796913/201612/796913-20161208132747460-313803974.png



b.使用ClientProxyFactoryBean类调用service服务

　　客户端必须加入cxf的jar包

　　浏览器访问<http://localhost:9090/ws?wsdl> ，查看service和operation

　　1) 不同项目中调用

[复制代码](javascript:void(0);)

package com.wp.test;

import com.wp.client.HelloWs;

import com.wp.client.HelloWsPortType;

public class Client {

public static void main(String[] args) {

HelloWs hw = new HelloWs();

HelloWsPortType h = hw.getHelloWsPort();

String result = h.sayHello("小强");

System.out.println(result);

}

}

[复制代码](javascript:void(0);)

　　1) 在同一项目中调用

[复制代码](javascript:void(0);)

package com.wp.service;

import org.apache.cxf.frontend.ClientProxyFactoryBean;

public class Client {

public static void main(String[] args) {

ClientProxyFactoryBean cfb = new ClientProxyFactoryBean();

cfb.setAddress("http://localhost:9090/ws");

cfb.setServiceClass(HelloWs.class);

HelloWs hw = (HelloWs) cfb.create();

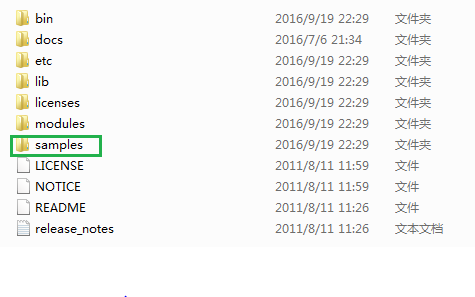
System.out.println(hw.sayHello("明明"));

}

}

[复制代码](javascript:void(0);)

类似的例子在apache-cxf-2.4.2安装包下的samples中，开发的时候可以查看



**三、使用JaxWsServerFactoryBean类发布服务**

**1.服务端**

[复制代码](javascript:void(0);)

package com.wp.service;

import javax.jws.WebParam;

import javax.jws.WebService;

@WebService

public interface HelloWs {

public String sayHello(@WebParam(name="text") String text);

}

[复制代码](javascript:void(0);)

[复制代码](javascript:void(0);)

package com.wp.service;

import javax.jws.WebService;

@WebService(endpointInterface = "com.wp.service.HelloWs")

public class HelloWsImpl implements HelloWs {

@Override

public String sayHello(String name) {

return "hello " + name;

}

}

[复制代码](javascript:void(0);)

[复制代码](javascript:void(0);)

package com.wp.service;

import org.apache.cxf.interceptor.LoggingInInterceptor;

import org.apache.cxf.interceptor.LoggingOutInterceptor;

import org.apache.cxf.jaxws.JaxWsServerFactoryBean;

public class Server {

public static void main(String[] args) {

JaxWsServerFactoryBean jsfb = new JaxWsServerFactoryBean();

//1.服务提供者实现的接口

jsfb.setServiceClass(HelloWs.class);

//2.指定访问路径

jsfb.setAddress("http://localhost:9090/ws");

//3.指定服务实现类

jsfb.setServiceBean(new HelloWsImpl());

//jsfb.getInInterceptors().add(new LoggingInInterceptor());

//jsfb.getOutInterceptors().add(new LoggingOutInterceptor());

//4.发布

jsfb.create();

System.out.println("发布成功...");

}

}

[复制代码](javascript:void(0);)

**2.客户端**

生成客户端代码：

https://images2015.cnblogs.com/blog/796913/201612/796913-20161208133616522-989214205.png

a.在不同项目中

[复制代码](javascript:void(0);)

package com.wp.test;

import org.apache.cxf.frontend.ClientProxyFactoryBean;

import org.apache.cxf.interceptor.LoggingInInterceptor;

import org.apache.cxf.interceptor.LoggingOutInterceptor;

import org.apache.cxf.jaxws.JaxWsProxyFactoryBean;

import com.wp.client.HelloWs;

public class Client {

public static void main(String[] args) {

JaxWsProxyFactoryBean factory = new JaxWsProxyFactoryBean();

// factory.getInInterceptors().add(new LoggingInInterceptor());

// factory.getOutInterceptors().add(new LoggingOutInterceptor());

factory.setAddress("http://localhost:9090/ws");

HelloWs client = factory.create(HelloWs.class);

System.out.println(client.sayHello("World"));

}

}

[复制代码](javascript:void(0);)

b.在同一项目中

[复制代码](javascript:void(0);)

package com.wp.service;

import org.apache.cxf.jaxws.JaxWsProxyFactoryBean;

public class Client {

public static void main(String[] args) {

JaxWsProxyFactoryBean bean = new JaxWsProxyFactoryBean();

bean.setAddress("http://localhost:9090/ws");

bean.setServiceClass(HelloWs.class);

HelloWs hw = (HelloWs) bean.create();

System.out.println(hw.sayHello("www"));

}

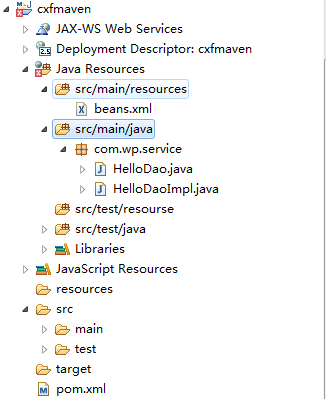
}

[复制代码](javascript:void(0);)

**四、spring和cxf的集成（MyEclipse）jaxws:server配置**

　　CXF原生支持spring，可以和Spring无缝集成。本例通过tomcat启动容器来启动cxf服务。

**1.新建一个maven项目（写过一个随笔），并且加入jar包**



https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4\_0\_0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.wp.test</groupId>

<artifactId>cxfmaven</artifactId>

<packaging>war</packaging>

<version>0.0.1-SNAPSHOT</version>

<name>cxfmaven Maven Webapp</name>

<url>http://maven.apache.org</url>

<properties>

<spring.version>3.2.5.RELEASE\_BUNDLE</spring.version>

</properties>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>3.8.1</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-asm</artifactId>

<version>3.0.5.RELEASE</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-beans</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-expression</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-tx</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>log4j</groupId>

<artifactId>log4j</artifactId>

<version>1.2.17</version>

</dependency>

<dependency>

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

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

<version>2.4.2</version>

</dependency>

<dependency>

<groupId>commons-logging</groupId>

<artifactId>commons-logging</artifactId>

<version>1.1</version>

</dependency>

<dependency>

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

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

<version>2.4.2</version>

</dependency>

<dependency>

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

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

<version>2.4.2</version>

</dependency>

<!-- Jetty is needed if you're using the CXFServlet -->

<dependency>

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

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

<version>2.4.2</version>

</dependency>

</dependencies>

<build>

<finalName>cxfmaven</finalName>

</build>

</project>

[复制代码](javascript:void(0);)

**2.让spring管理ServerFactoryBean**

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

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

<beans

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

xmlns:jaxws="http://cxf.apache.org/jaxws"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

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

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

">

<!--

1 导入cxf发布的命名空间

xmlns:jaxws="http://cxf.apache.org/jaxws"

http://cxf.apache.org/jaxws

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

2 通过配置发布webservice服务

2.1 发布简单的webservice服务(不需要定义服务器的接口 ，了解)

<jaxws:endpoint address="" implementor="">

</jaxws:endpoint>

2.2 发布标准的webservice服务

2.2.1 发布的地址 address

2.2.2 发布的服务器实现的接口 serviceClass

2.2.3 指定服务具体提供者（实现类对象） jaxws:serviceBean

-->

<!-- 实例化一个JaxWsServerFactoryBean对象 -->

<jaxws:server address="/helloWs"

serviceClass="com.wp.service.HelloWs">

<jaxws:serviceBean>

<bean class="com.wp.service.HelloWsImpl"></bean>

</jaxws:serviceBean>

</jaxws:server>

</beans>

[复制代码](javascript:void(0);)

**3.cxf集成到web容器中**

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

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

<web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://java.sun.com/xml/ns/javaee

http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd">

<display-name></display-name>

<welcome-file-list>

<welcome-file>index.jsp</welcome-file>

</welcome-file-list>

<servlet>

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

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

<!-- 初始化CXFServlet -->

<init-param>

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

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

</init-param>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

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

<url-pattern>/\*</url-pattern>

</servlet-mapping>

</web-app>

[复制代码](javascript:void(0);)

**4.启动web容器发布webservice服务**

wsdl访问地址：http://localhost:8080/wsService/helloWs?wsdl

1. 通过Eclipse中的Web Services Explorer进行测试；
2. 建立一个客户端项目进行测试，跟上面一样。