# word

# 1 CXF

## CXF概述

* CXF是Apache旗下的，所以全称叫Apache CXF
* Apache CXF = Celtix + Xfire，开始叫 Apache CeltiXfire，后来更名为 Apache CXF 了，以下简称为 CXF。
* **Apache CXF 是一个开源的web Services 框架**
* CXF 帮助您构建和开发 web Services
* 它支持多种协议，比如：SOAP1.1,1,2、XML/HTTP、RESTful HTTP 或者 CORBA。
* CORBA（Common Object Request Broker Architecture公共对象请求代理体系结构,早期语言使用的WS。C,c++,C#）
* Cxf是基于SOA总线结构，依靠spring完成模块的集成，实现SOA方式。
* CXF可以灵活的部署：可以运行有Tomcat,Jboss,Jetty(内置),weblogic上面。

## 1.2 CXF下载

* 访问http://cxf.apache.org/download.html官网下载
* 解压zip后，里面有个bin目录和lib目录
  + bin目录是一些命令，如生成客户端代码的命令(**wsdl2java**)
  + lib目录是我们在项目中要使用的jar包

|  |
| --- |
|  |
|  |
|  |

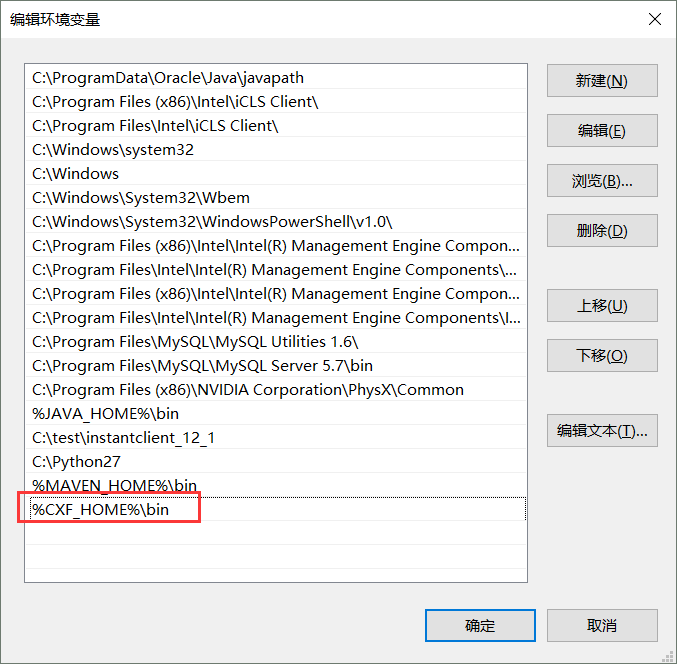
## 1.3 CXF环境变量配置

### window配置

用户/系统变量 JAVA\_HOME=C:\Program Files\Java\jdk1.8.0\_121

用户/系统变量 CXF\_HOME= C:\apache-cxf-2.7.11

系统变量 Path = %JAVA\_HOME%\bin;%CXF\_HOME%\bin;



## 1.4 CXF的第一个例子

### 1.4.1编写服务端代码

**步骤**

* 创建一个java工程
* 导入cxf的jar包
* 只有导入一个cxf-manifest.jar就可以，内部会导入所有jar【IEDA就不要理】
* 创建天气服务接口和实现类
* **在接口上添加@webservice**(与jax-ws不同，它是在实现类上添加的)

|  |
| --- |
|  |
|  |

**发布webservice服务**:

使用JaxWsServerFactoryBean 来发布（soap12）

加入拦截器,拦截请求

bean.getInInterceptors().add(new LoggingInInterceptor());

bean.getOutInterceptors().add(new LoggingOutInterceptor());

|  |
| --- |
| **import** org.apache.cxf.jaxws.JaxWsServerFactoryBean;  **public** **class** WeatherServer {  **public** **static** **void** main(String[] args) {  //创建服务工厂Bean  JaxWsServerFactoryBean factoryBean = **new** JaxWsServerFactoryBean();  //设置服务实现对象  factoryBean.setServiceBean(**new** WeatherIntefaceImpl());  //设置服务发布地址  factoryBean.setAddress("http://127.0.0.1:12345/weather");  //创建服务  factoryBean.create();  }  } |

### 14.2 wsimport生成客户端面代码

命令：wsimport -s . http://127.0.0.1:12345/weather?wsdl

### 14.3 wsdl2java生成客户端面代码

命令：wsdl2java –d . [http://127.0.0.1:12345/weather?wsdl](http://192.168.1.100:1234/weather?wsdl)

* **wsdl2java在windodw jdk1.8的执行失败原因**
* **Windodw jdk1.8的版本，使用wsdl2java未能成功**
* **因为jre和jdk都用的是jdk8, 而刚好jaxb-xjc-2.2.7对jdk8有这个bug**
* **解决办法**

|  |  |
| --- | --- |
| |  | | --- | | **Create a file named jaxp.properties (if it doesn't exist) under /path/to/jdk1.8.0/jre/lib and then write this line in it: javax.xml.accessExternalSchema = all** | |

### 1.4.4编写客户端代码

掌握两种方式：JAX-WS和CXF的方式

|  |
| --- |
|  |
| **package** com.gyf.test;  **import** org.apache.cxf.jaxws.JaxWsProxyFactoryBean;  **import** com.gyf.weather.WeatherInteface;  **public** **class** Demo{  **public** **static** **void** main(String[] args) {  //创建代理工厂  JaxWsProxyFactoryBean factoryBean = **new** JaxWsProxyFactoryBean();    //设置 服务接口类型  factoryBean.setServiceClass(WeatherInteface.**class**);    //设置服务器地址  factoryBean.setAddress("http://127.0.0.1:12345/weather?wsdl");    WeatherInteface weather = factoryBean.create(WeatherInteface.**class**);    System.***out***.println(weather.queryWeather("北京"));  }  } |

### 1.4.5如何生成soap1.2的webservice

只需要在接口上添加BindingType即可，如下图



# CXF+Spring整合发布SOAP服务

掌握如何通过CXF+Spring发布SOAP协议的服务

## 2.1 服务端开发步骤

### 创建Web程序

名称为【07.spring\_cxf\_web】

### 将cxf的lib放在WEB-INF/lib中

把【springmvc3.1.4与cxf2.7.11整合jar】目的jar都导入

### 创建SEI接口和实现类

把前面写的copy过来即可

|  |
| --- |
|  |

### 配置spring配置文件，添加applicationContext.xml （内容）

|  |
| --- |
| <?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:jaxrs=*"http://cxf.apache.org/jaxrs"* xmlns:cxf=*"http://cxf.apache.org/core"*  xsi:schemaLocation=*"http://www.springframework.org/schema/beans*  *http://www.springframework.org/schema/beans/spring-beans.xsd*  *http://cxf.apache.org/jaxrs http://cxf.apache.org/schemas/jaxrs.xsd*  *http://cxf.apache.org/jaxws http://cxf.apache.org/schemas/jaxws.xsd*  *http://cxf.apache.org/core http://cxf.apache.org/schemas/core.xsd"*> </beans> |

### 在applicationContext.xml配置webservice服务内容

|  |
| --- |
| <!--1.配置个服务接口实现类 -->  <bean id=*"weatherInterfaceImpl"* class=*"com.gyf.weather.WeatherInterfaceImpl"*></bean>    <!--2.配置cxf服务工厂bean -->  <jaxws:server address=*"/weather"* serviceClass=*"com.gyf.weather.WeatherInteface"*>  <jaxws:serviceBean>  <ref bean=*"weatherInterfaceImpl"*/>  </jaxws:serviceBean>  </jaxws:server> |

### 配置web.xml加载spring,和CXF请求拦截

|  |
| --- |
| <context-param>  <!--contextConfigLocation是不能修改的 -->  <param-name>contextConfigLocation</param-name>  <param-value>classpath:applicationContext.xml</param-value>  </context-param>  <listener>  <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>  </listener>  <!-- 2.配置CXF -->  <servlet>  <servlet-name>CXF</servlet-name>  <servlet-class>org.apache.cxf.transport.servlet.CXFServlet</servlet-class>  </servlet>  <servlet-mapping>  <servlet-name>CXF</servlet-name>  <url-pattern>/ws/\*</url-pattern>  </servlet-mapping> |

### 布置到tomcat中，启动tomcat

### 在浏览器访问

http://127.0.0.1:8080/09.cxf\_spring\_web/ws/weather?wsdl

|  |
| --- |
|  |

## 2.2 配置日志拦截器

步骤

在applicationContext.xml配置拦截器

**http://localhost:8080/项目名称/ws可以看到所有服务列表**

|  |
| --- |
| <!--1.配置个服务接口实现类 -->  <bean id=*"weatherInterfaceImpl"* class=*"com.gyf.weather.WeatherInterfaceImpl"*></bean>    <!--2.配置cxf服务工厂bean -->  <jaxws:server address=*"/weather"* serviceClass=*"com.gyf.weather.WeatherInterface"*>  <jaxws:serviceBean>  <ref bean=*"weatherInterfaceImpl"*/>  </jaxws:serviceBean>    <jaxws:inInterceptors>  <ref bean=*"inInterceptor"*/>  </jaxws:inInterceptors>    <jaxws:outInterceptors>  <ref bean=*"outInterceptor"*/>  </jaxws:outInterceptors>  </jaxws:server>    <!-- 3.配置cxf的日志拦截器 -->  <bean name=*"inInterceptor"* class=*"org.apache.cxf.interceptor.LoggingInInterceptor"*/>  <bean name=*"outInterceptor"* class=*"org.apache.cxf.interceptor.LoggingOutInterceptor"*/> |
|  |

## 2.3 Endpoint标签发布服务[喜欢这种，方便]

步骤

1. 在服务端项目中添加一个任意服务

|  |
| --- |
|  |

1. 在applicationContext.xml配置endpoint

|  |
| --- |
|  |

## 2.4 客户端开发

步骤

* 创建一个javaweb项目
* 导入cxf的jar包，(内部已经有spring的jar包了)
* 生成客户端面代码

wsdl2java -d . http://127.0.0.1:8080/07.spring\_cxf\_web/ws/weather?wsdl

* 配置applicationContext.xml

|  |
| --- |
|  |

* 写一个main方法来调用接口

|  |
| --- |
| **public** **static** **void** main(String[] args) {  ApplicationContext context = **new** ClassPathXmlApplicationContext("classpath:applicationContext.xml");  WeatherInterface weatherService = (WeatherInterface) context.getBean("weatherService");    System.***out***.println(weatherService.queryWeather("广州"));  } |

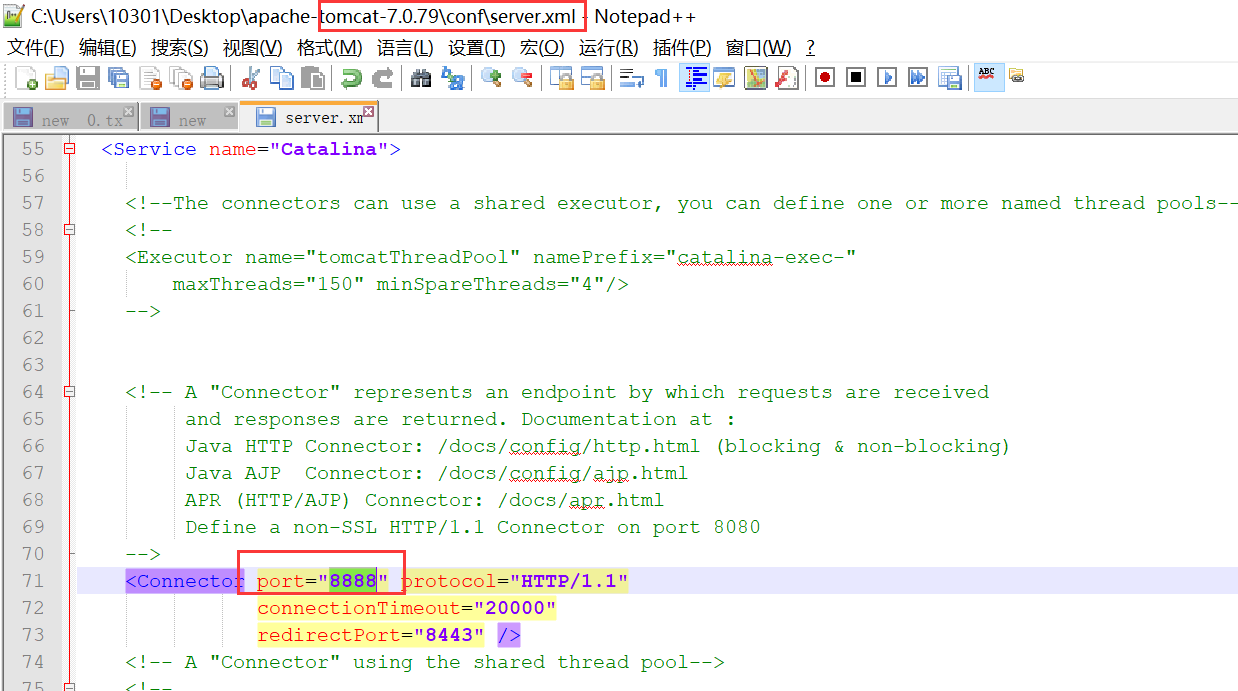
* JSP-查询天气



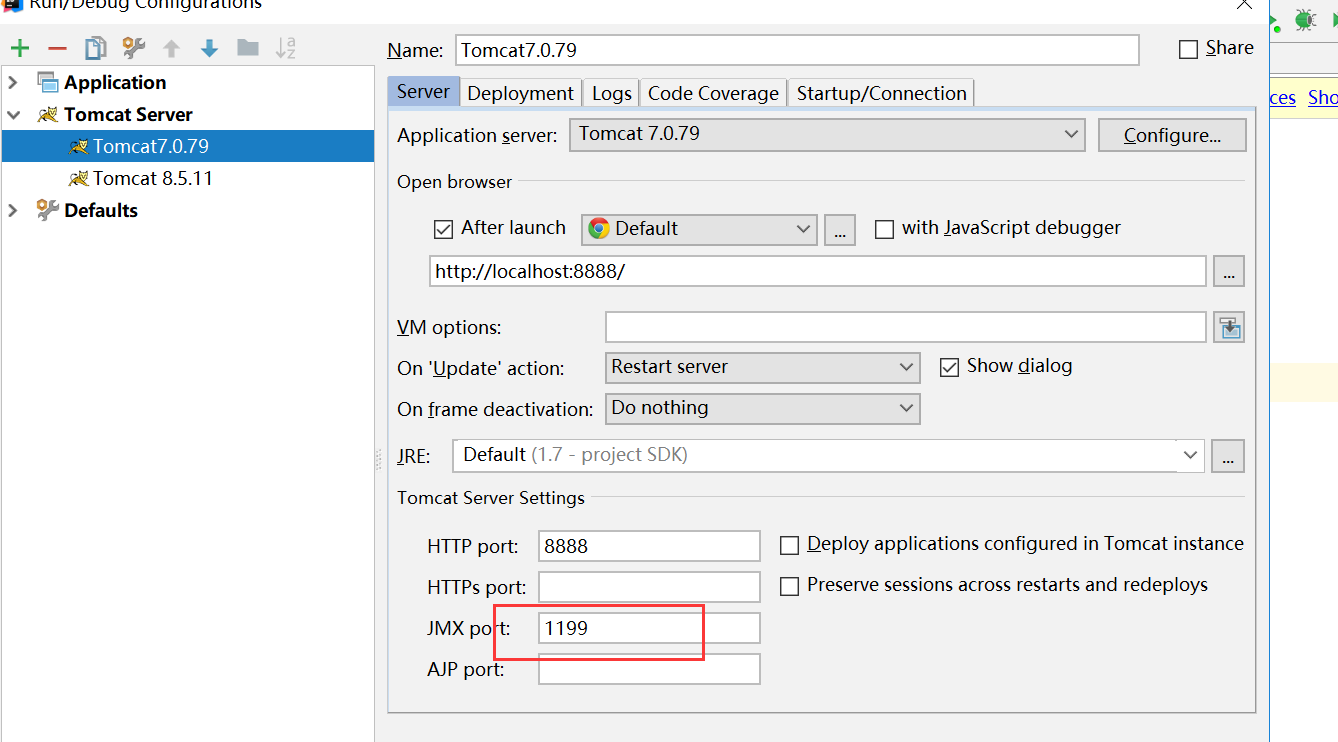
* 写Servlet

|  |
| --- |
| @WebServlet(**"/WeatherSearchServlet"**) **public class** WeatherSearchServlet **extends** HttpServlet {   **protected void** doPost(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {  *//解决post请求乱码* request.setCharacterEncoding(**"UTF-8"**);  *//1.获取参数* String cityName = request.getParameter(**"cityName"**);  System.***out***.println(cityName);   *//2.获取weatherService,【在web.xml中加载spring配置文件】* ApplicationContext context = WebApplicationContextUtils.*getWebApplicationContext*(**this**.getServletContext());  WeatherInterface weatherService = (WeatherInterface) context.getBean(**"weatherService"**);    *//3.调用weatherService* String weatherInfo = weatherService.queryWeather(cityName);    *//4.把数据存在request* request.setAttribute(**"weatherInfo"**,weatherInfo);   *//5.返回一个jsp页面，在jsp页面显示数据* request.getRequestDispatcher(**"/index.jsp"**).forward(request,response);  } |

* 用tomcat8来跑服务端，用tomcat7跑客户端面，注意：tomcat8使用8080端面的话，那tomcat7就需要改端口，改端口在server.xml文件中为8888



* 改在IEDA配置tomcat7时的1099端口



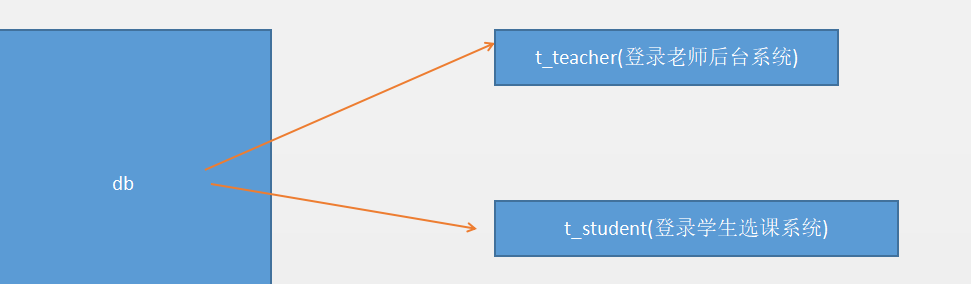
# CXF发布REST服务

## 3.1 REST概述

* REST 是一种**软件架构模式，只是一种风格**
* rest服务采用HTTP 做传输协议
* REST 对于HTTP 的利用**实现精确的资源定位**
* rest要求对资源定位更加准确，如下：
  + 非rest方式：

http://ip:port/queryUser.action?userType=student&id=001

http://ip:port/queryUser.action?userType=teacher&id=002



* + Rest方式：

http://ip:port/user/student/query/001

http://ip:port/user/teacher/query/001

* Rest方式表示**互联网上的资源更加准确，但是也有缺点，可能目录的层级较多不容易理解**
* REST 是一种**软件架构理念**，现在被移植到Web 服务上，那么在开发Web 服务上，偏于面向资源的服务适用于REST，REST 简单易用，效率高，SOAP 成熟度较高，安全性较好。
* **注意：REST 不等于WebService-soap，JAX-RS 只是将REST 设计风格应用到Web 服务开发上**。

## 3.2案例

* 发布查询学生信息的服务，**以json和xml数据**格式返回。

**功能：**

1. 通过ID查询一个学生信息服务
2. 查询所有学生信息服务

## 3.3案例步骤

### A.创建Java程序【09.cxf\_rest\_server】

### B.导入cxf 的jar包

### C.创建学生pojo类,添加注解

|  |
| --- |
|  |

### D.创建SEI和SEI实现类,添加注解

**Service Endpoint Interface 发布的服务接口**

在接口名上面声明**@WebService**和**@Path**

在接口方法名上面声明@GET/@POST、@Path、@Produces

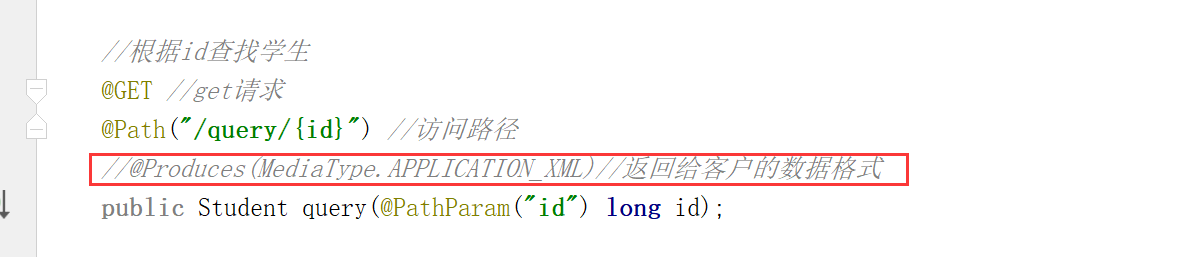
|  |
| --- |
| @WebService  @Path("student")  **public** **interface** StudentInterface {    @GET//指定请求方式  @Produces(MediaType.***APPLICATION\_XML***)//指定返回数据格式是xml还是json  @Path("/query/{id}")//指定访问路径,query映射到方法，id映射到参数  **public** Student query(@PathParam("id")**long** id);  @GET  @Produces(MediaType.***APPLICATION\_XML***)  @Path("/queryList/{name}")  **public** List<Student> queryList(@PathParam("name")String name);  } |
|  |

### E.程序代码发布

|  |
| --- |
| **public** **class** Server {  **public** **static** **void** main(String[] args) {  JAXRSServerFactoryBean service = **new** JAXRSServerFactoryBean();  //设置服务接口  service.setResourceClasses(StudentInterfaceImpl.**class**);  //设置服务实现  service.setServiceBean(**new** StudentInterfaceImpl());  //设置地址  service.setAddress("http://127.0.0.1:12345/user");  //发布服务  service.create();  }  } |

### F.测试发布,切换xml,json返回类型

* http://127.0.0.1:12345/user/student/query/23
* http://127.0.0.1:12345/user/student/queryList/zs
* **如果不写@Produces,可以在访问路径后面添加?\_type=xml/json来实现数据格式转换**



## 3.4客户端REST的GET请求

客户端写REST的GET请求,只需要使用HttpURLConnection访问资源路径即可

### 代码如下:

|  |
| --- |
| //创建url  URL url = new URL("http://127.0.0.1:12345/user/student/query/23?\_type=json");    //打开连接  HttpURLConnection con = (HttpURLConnection) url.openConnection();    //设置请求方式  con.setRequestMethod("GET");  /\*\*默认connection没有读写的权限\*/  con.setDoOutput(true);    //获取响应  int code = con.getResponseCode();  if(code == 200){  System.out.println("响应成功");  //读取数据  InputStream is = con.getInputStream();  InputStreamReader isr = new InputStreamReader(is);  BufferedReader br = new BufferedReader(isr);  String line = null;  while((line = br.readLine()) != null){  System.out.println(line);  }  br.close();  isr.close();  is.close();  }else{  System.out.println("响应失败");  } |

## 3.5 CXF+Spring web整合发布REST服务

关键点就是使用jaxrs:server来发布rest服务

