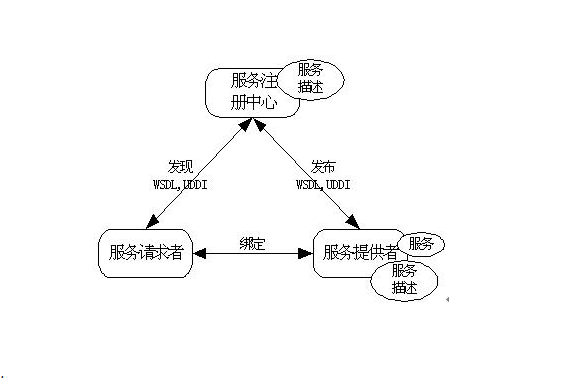
## Webservice基本概念

Web Service也叫XML Web Service WebService是一种可以接收从Internet或者Intranet上的其它系统中传递过来的请求，轻量级的独立的通讯技术。是:通过SOAP在Web上提供的软件服务，使用WSDL文件进行说明，并通过UDDI进行注册。  
  
XML：(Extensible Markup Language)扩展型可标记语言。面向短期的临时数据处理、面向万维网络，是Soap的基础。  
  
Soap：(Simple Object Access Protocol)简单对象存取协议。是XML Web Service 的通信协议。当用户通过UDDI找到你的WSDL描述文档后，他通过可以SOAP调用你建立的Web服务中的一个或多个操作。SOAP是XML文档形式的调用方法的规范，它可以支持不同的底层接口，像HTTP(S)或者SMTP。  
  
WSDL：(Web Services Description Language) WSDL 文件是一个 XML 文档，用于说明一组 SOAP 消息以及如何交换这些消息。大多数情况下由软件自动生成和使用。  
  
UDDI (Universal Description, Discovery, and Integration) 是一个主要针对Web服务供应商和使用者的新项目。在用户能够调用Web服务之前，必须确定这个服务内包含哪些商务方法，找到被调用的接口定义，还要在服务端来编制软件，UDDI是一种根据描述文档来引导系统查找相应服务的机制。UDDI利用SOAP消息机制（标准的XML/HTTP）来发布，编辑，浏览以及查找注册信息。它采用XML格式来封装各种不同类型的数据，并且发送到注册中心或者由注册中心来返回需要的数据。

## 调用原理



实现一个完整的Web服务包括以下步骤：  
  
◆ Web服务提供者设计实现Web服务，并将调试正确后的Web服务通过Web服务中介者发布，并在UDDI注册中心注册； （发布）  
  
◆ Web服务请求者向Web服务中介者请求特定的服务，中介者根据请求查询UDDI注册中心，为请求者寻找满足请求的服务； （发现）  
  
◆ Web服务中介者向Web服务请求者返回满足条件的Web服务描述信息，该描述信息用WSDL写成，各种支持Web服务的机器都能阅读；（发现）  
  
◆ 利用从Web服务中介者返回的描述信息生成相应的SOAP消息，发送给Web服务提供者，以实现Web服务的调用；（绑定）  
  
◆ Web服务提供者按SOAP消息执行相应的Web服务，并将服务结果返回给Web服务请求者。（绑定）

## 三．webservice简单的例子

1. 创建一个web项目
2. 在web项目中创建一个类HelloWorld.java的类



1. 运行main方法，在浏览器中输入http://localhost:8080/adfadfa?wsdl，出现一下说明发布成功：



## [CXF提供Client调用WebService接口的方法](http://blog.csdn.net/a128953ad/article/details/49905089)

1、 JaxWsProxyFactoryBean

   简介：调用方式采用了和RMI类似的机制，即客户端直接服务器端提供的服务接口(interface),CXF通过运行时代理生成远程服务的代理对象，在客户端完成对webservice的访问;几个必填的字段：setAddress-这个就是我们发布webservice时候的地址，保持一致

  缺点：这种调用service的好处在于调用过程非常简单，就几行代码就完成一个webservice的调用，但是客户端也必须依赖服务器端的接口，这种调用方式限制是很大的，要求服务器端的webservice必须是java实现--这样也就失去了使用webservice的意义

1. **import org.apache.cxf.jaxws.JaxWsProxyFactoryBean;**
2. **public** **class** Client {
3. **public** **static** **void** main(String[] args) {
4. JaxWsProxyFactoryBean bean = **new** JaxWsProxyFactoryBean();
5. bean.setServiceClass(HelloWorldService.**class**);
6. bean.setAddress("http://localhost:9090/helloWorldService");
7. HelloWorldService helloWorldService = (HelloWorldService)bean.create();
8. String result = helloWorldService.sayHello("Kevin");
9. System.out.println(result);
10. }

2、JaxWsDynamicClientFactory  [Dynamci：动态的]

     简介：只要指定服务器端wsdl文件的位置，然后指定要调用的方法和方法的参数即可，不关心服务端的实现方式。

           wsdl [Web Services Description Language]网络服务描述语言是[Web Service](http://baike.baidu.com/view/67105.htm" \t "http://blog.csdn.net/a128953ad/article/details/_blank)的描述语言，它包含一系列描述某个web service的定义

1. **import org.apache.cxf.jaxws.endpoint.dynamic.JaxWsDynamicClientFactory;**
2. **public** **class** Client2 {
3. **public** **static** **void** main(String[] args) **throws** Exception {
4. JaxWsDynamicClientFactory clientFactory = JaxWsDynamicClientFactory.newInstance();
5. Client client = clientFactory.createClient("http://localhost:9090/helloWorldService?wsdl");
6. Object[] result = client.invoke("sayHello", "KEVIN");
7. System.out.println(result[0]);
8. }
9. }

3、JaxWsServerFactoryBean

     用JaxWsServerFactoryBean发布，需要独立的jetty包。

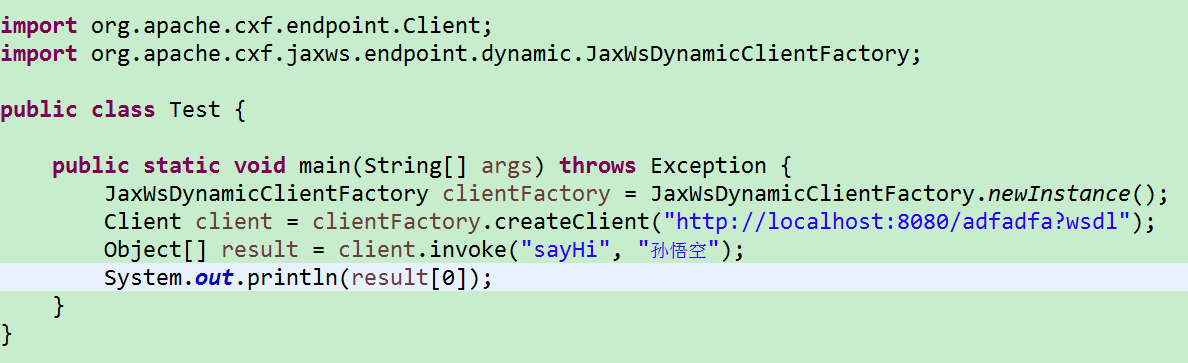
注：具体可以参考CXF API  ----http://cxf.apache.org/docs/dynamic-clients.html

## 调用webservice接口

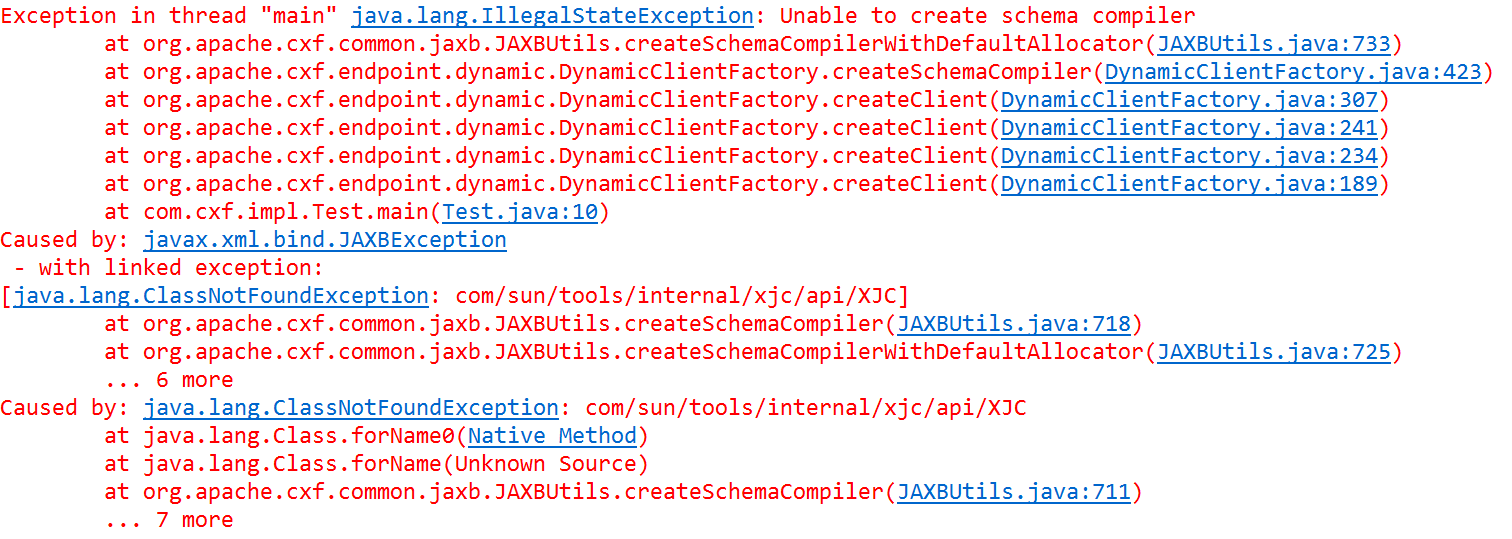
1. 在创建一个web项目
2. 导入cxf的jar包，这里用maven的依赖：

|  |
| --- |
| <!-- cxf -->  <dependency>  <groupId>org.apache.cxf</groupId>  <artifactId>cxf-rt-frontend-jaxws</artifactId>  <version>3.1.6</version>  </dependency>  <!-- https://mvnrepository.com/artifact/org.apache.cxf/cxf-core -->  <dependency>  <groupId>org.apache.cxf</groupId>  <artifactId>cxf-core</artifactId>  <version>3.1.6</version>  </dependency>  <!-- https://mvnrepository.com/artifact/org.apache.cxf/cxf-rt-transports-http -->  <dependency>  <groupId>org.apache.cxf</groupId>  <artifactId>cxf-rt-transports-http</artifactId>  <version>3.1.6</version>  </dependency> |

1. 创建测试类调用接口：



1. 运行发现如果出现一下错误



原因是：没有正确的使用jdk，在jdk的lib下面没有找到类

解决方法：使用正确的jdk

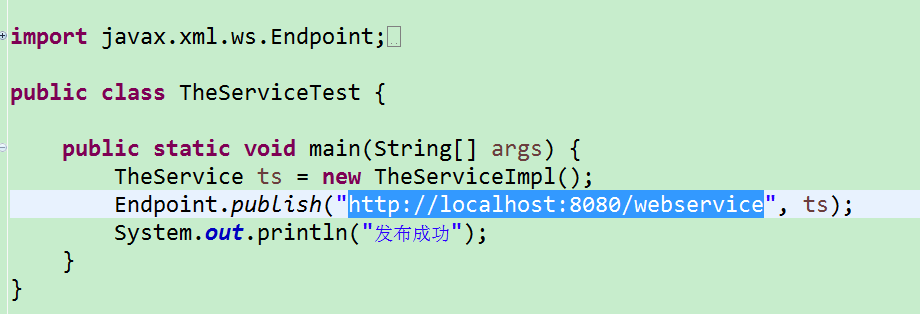
## 六．添加javabean

1. 服务类

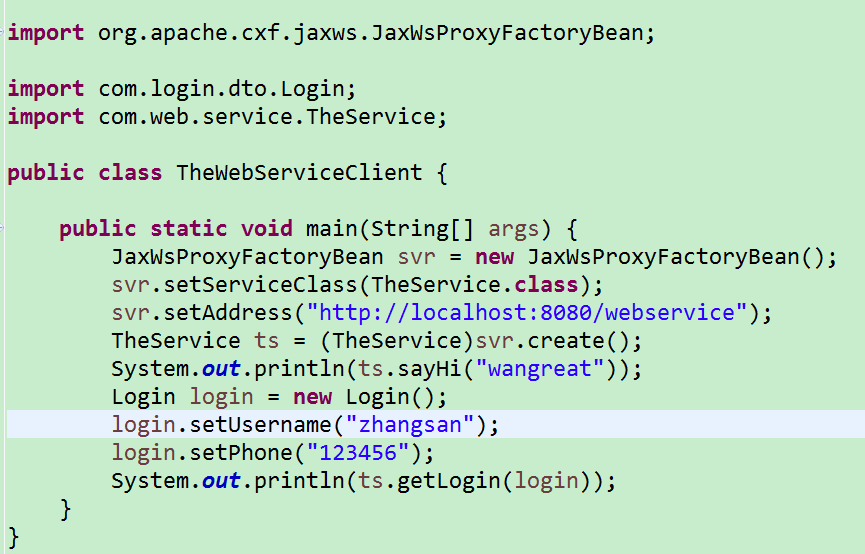




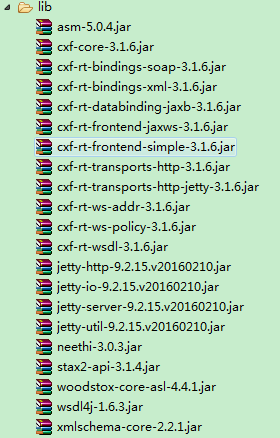
1. 服务发布类



1. 客户端调用类



1. 所用到的jar包：

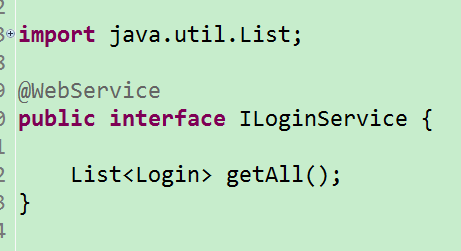


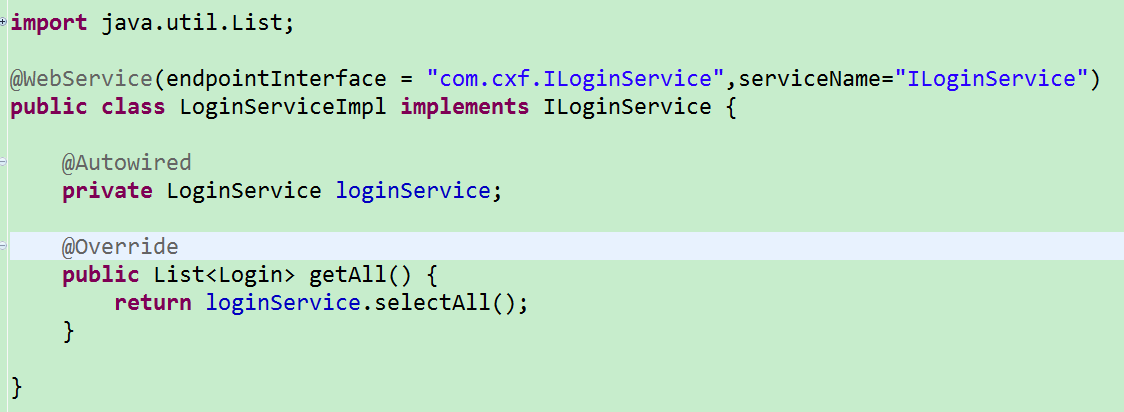
## 七．整合spring

1)添加相应的jar包

|  |
| --- |
| 1. **<project** xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" 2. xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"**>** 3. **<modelVersion>**4.0.0**</modelVersion>** 4. **<groupId>**com.neusoft.icelake**</groupId>** 5. **<artifactId>**restservice**</artifactId>** 6. **<version>**0.0.1-SNAPSHOT**</version>** 7. **<packaging>**war**</packaging>** 8. **<name>**restservice**</name>** 10. **<properties>** 11. **<project.build.sourceEncoding>**UTF-8**</project.build.sourceEncoding>** 12. **<java.version>**1.6**</java.version>** 13. **<spring.version>**3.2.8.RELEASE**</spring.version>** 14. **<servlet-api.version>**2.5**</servlet-api.version>** 15. **<slf4j.version>**1.7.5**</slf4j.version>** 16. **<cxf.version>**2.7.12**</cxf.version>** 17. **<jackson.version>**2.1.4**</jackson.version>** 19. **<mybatis.version>**3.2.8**</mybatis.version>** 20. **<mybatisspring.version>**1.2.2**</mybatisspring.version>** 21. **<druid.version>**1.0.11**</druid.version>** 22. **</properties>** 24. **<dependencies>** 25. <!-- spring begin --> 26. **<dependency>** 27. **<groupId>**org.springframework**</groupId>** 28. **<artifactId>**spring-core**</artifactId>** 29. **<version>**${spring.version}**</version>** 30. **</dependency>** 31. **<dependency>** 32. **<groupId>**org.springframework**</groupId>** 33. **<artifactId>**spring-beans**</artifactId>** 34. **<version>**${spring.version}**</version>** 35. **</dependency>** 36. **<dependency>** 37. **<groupId>**org.springframework**</groupId>** 38. **<artifactId>**spring-orm**</artifactId>** 39. **<version>**${spring.version}**</version>** 40. **</dependency>** 41. **<dependency>** 42. **<groupId>**org.springframework**</groupId>** 43. **<artifactId>**spring-context**</artifactId>** 44. **<version>**${spring.version}**</version>** 45. **</dependency>** 46. **<dependency>** 47. **<groupId>**org.springframework**</groupId>** 48. **<artifactId>**spring-webmvc-portlet**</artifactId>** 49. **<version>**${spring.version}**</version>** 50. **</dependency>** 51. **<dependency>** 52. **<groupId>**org.springframework**</groupId>** 53. **<artifactId>**spring-webmvc**</artifactId>** 54. **<version>**${spring.version}**</version>** 55. **</dependency>** 56. **<dependency>** 57. **<groupId>**org.springframework**</groupId>** 58. **<artifactId>**spring-web**</artifactId>** 59. **<version>**${spring.version}**</version>** 60. **</dependency>** 61. <!-- spring end --> 63. <!-- @Inject begin --> 64. **<dependency>** 65. **<groupId>**javax.inject**</groupId>** 66. **<artifactId>**javax.inject**</artifactId>** 67. **<version>**1**</version>** 68. **</dependency>** 69. <!-- @Inject end --> 71. <!-- servlet begin --> 72. **<dependency>** 73. **<groupId>**javax.servlet**</groupId>** 74. **<artifactId>**servlet-api**</artifactId>** 75. **<version>**${servlet-api.version}**</version>** 76. **</dependency>** 77. <!-- servlet end --> 79. <!-- cxf begin -->           <dependency>  <groupId>org.apache.cxf</groupId>  <artifactId>cxf-rt-frontend-jaxws</artifactId>  <version>3.1.6</version>  </dependency>  <!-- https://mvnrepository.com/artifact/org.apache.cxf/cxf-core -->  <!-- 这个包是必须的，因为会使用拦截器等记录时，需要这个包 -->  <dependency>  <groupId>org.apache.cxf</groupId>  <artifactId>cxf-core</artifactId>  <version>3.1.6</version>  </dependency>  <!-- https://mvnrepository.com/artifact/org.apache.cxf/cxf-rt-transports-http -->  <dependency>  <groupId>org.apache.cxf</groupId>  <artifactId>cxf-rt-transports-http</artifactId>  <version>3.1.6</version>  </dependency>  <dependency>  <groupId>org.apache.cxf</groupId>  <artifactId>cxf-rt-transports-http-jetty</artifactId>  <version>3.1.6</version>  </dependency>  <!-- https://mvnrepository.com/artifact/org.apache.cxf/cxf-rt-ws-security -->  <dependency>  <groupId>org.apache.cxf</groupId>  <artifactId>cxf-rt-ws-security</artifactId>  <version>3.1.6</version>   1. </dependency> 2. <!-- cxf end --> 4. <!-- JSON begin --> 5. **<dependency>** 6. **<groupId>**com.fasterxml.jackson.core**</groupId>** 7. **<artifactId>**jackson-core**</artifactId>** 8. **<version>**${jackson.version}**</version>** 9. **</dependency>** 10. **<dependency>** 11. **<groupId>**com.fasterxml.jackson.core**</groupId>** 12. **<artifactId>**jackson-databind**</artifactId>** 13. **<version>**${jackson.version}**</version>** 14. **</dependency>** 15. **<dependency>** 16. **<groupId>**com.fasterxml.jackson.core**</groupId>** 17. **<artifactId>**jackson-annotations**</artifactId>** 18. **<version>**${jackson.version}**</version>** 19. **</dependency>** 20. **<dependency>** 21. **<groupId>**com.fasterxml.jackson.module**</groupId>** 22. **<artifactId>**jackson-module-jaxb-annotations**</artifactId>** 23. **<version>**${jackson.version}**</version>** 24. **</dependency>** 25. **<dependency>** 26. **<groupId>**com.fasterxml.jackson.jaxrs**</groupId>** 27. **<artifactId>**jackson-jaxrs-json-provider**</artifactId>** 28. **<version>**${jackson.version}**</version>** 29. **</dependency>** 30. **<dependency>** 31. **<groupId>**org.codehaus.jackson**</groupId>** 32. **<artifactId>**jackson-mapper-asl**</artifactId>** 33. **<version>**1.9.0**</version>** 34. **</dependency>** 35. **<dependency>** 36. **<groupId>**org.codehaus.jackson**</groupId>** 37. **<artifactId>**jackson-core-asl**</artifactId>** 38. **<version>**1.9.0**</version>** 39. **</dependency>** 40. <!-- JSON end --> 42. <!-- MyBatis --> 43. **<dependency>** 44. **<groupId>**org.mybatis**</groupId>** 45. **<artifactId>**mybatis**</artifactId>** 46. **<version>**${mybatis.version}**</version>** 47. **</dependency>** 48. **<dependency>** 49. **<groupId>**org.mybatis**</groupId>** 50. **<artifactId>**mybatis-spring**</artifactId>** 51. **<version>**${mybatisspring.version}**</version>** 52. **</dependency>** 53. <!-- AOP begin --> 54. **<dependency>** 55. **<groupId>**org.aspectj**</groupId>** 56. **<artifactId>**aspectjrt**</artifactId>** 57. **<version>**1.7.4**</version>** 58. **</dependency>** 59. **<dependency>** 60. **<groupId>**org.aspectj**</groupId>** 61. **<artifactId>**aspectjweaver**</artifactId>** 62. **<version>**1.7.4**</version>** 63. **</dependency>** 64. **<dependency>** 65. **<groupId>**cglib**</groupId>** 66. **<artifactId>**cglib**</artifactId>** 67. **<version>**3.1**</version>** 68. **</dependency>** 69. <!-- AOP end --> 70. <!-- jdbc driver --> 71. **<dependency>** 72. **<groupId>**localrep.sybase**</groupId>** 73. **<artifactId>**jconn**</artifactId>** 74. **<version>**2.0**</version>** 75. **</dependency>** 76. <!-- connection pool --> 77. **<dependency>** 78. **<groupId>**com.alibaba**</groupId>** 79. **<artifactId>**druid**</artifactId>** 80. **<version>**${druid.version}**</version>** 81. **</dependency>** 82. **</dependencies>** 84. **<build>** 85. **<plugins>** 86. <!-- compiler插件, 设定JDK版本 --> 87. **<plugin>** 88. **<groupId>**org.apache.maven.plugins**</groupId>** 89. **<artifactId>**maven-compiler-plugin**</artifactId>** 90. **<version>**3.0**</version>** 91. **<configuration>** 92. **<source>**${java-version}**</source>** 93. **<target>**${java-version}**</target>** 94. **</configuration>** 95. **</plugin>** 96. **</plugins>** 97. **</build>** 99. <!-- PROGRAMER INFO --> 100. **<organization>** 101. **<name>**Neusoft**</name>** 102. **<url>**http://www.neusoft.com**</url>** 103. **</organization>** 104. **</project>** |

1. 实现服务





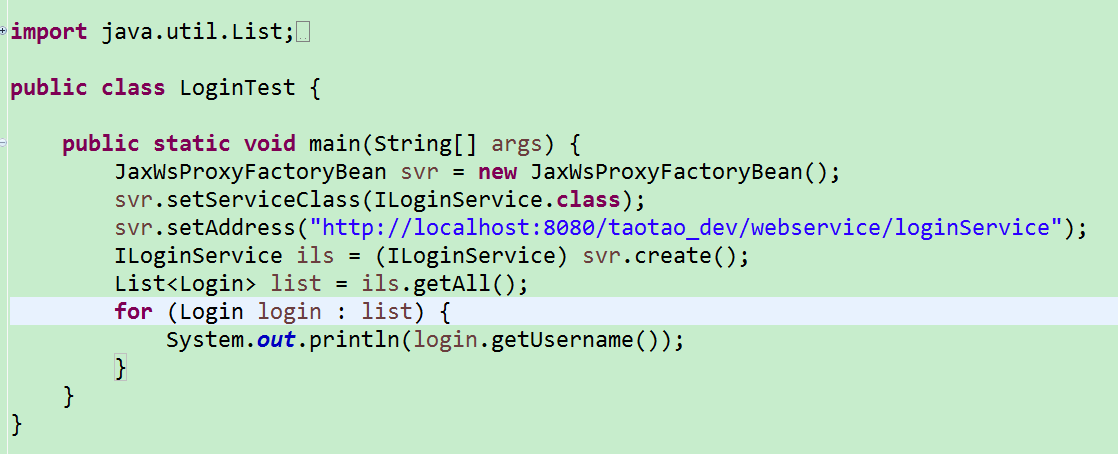
1. 配置applicationContext-cxf.xml

|  |
| --- |
| <beans xmlns=*"http://www.springframework.org/schema/beans"*  xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xmlns:p=*"http://www.springframework.org/schema/p"*  xmlns:aop=*"http://www.springframework.org/schema/aop"*  xmlns:context=*"http://www.springframework.org/schema/context"*  xmlns:jee=*"http://www.springframework.org/schema/jee"*  xmlns:tx=*"http://www.springframework.org/schema/tx"*  xmlns:jaxws=*"http://cxf.apache.org/jaxws"*  xsi:schemaLocation=*"*  *http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-4.0.xsd*  *http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-4.0.xsd*  *http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.0.xsd*  *http://www.springframework.org/schema/jee http://www.springframework.org/schema/jee/spring-jee-4.0.xsd*  *http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-4.0.xsd*  *http://cxf.apache.org/jaxws http://cxf.apache.org/schemas/jaxws.xsd"*>  <import resource=*"classpath:META-INF/cxf/cxf.xml"*/>  <import resource=*"classpath:META-INF/cxf/cxf-servlet.xml"*/>  <bean id=*"helloWorld"* class=*"com.cxf.impl.HelloWorldImpl"*></bean>  <!-- 定义webservice的发布接口 -->  <jaxws:endpoint implementor=*"#helloWorld"* address=*"/HelloWorld"*>    </jaxws:endpoint>    <bean id=*"loginService"* class=*"com.cxf.impl.LoginServiceImpl"*/>  <jaxws:endpoint implementor=*"#loginService"* address=*"/loginService"*/>  </beans> |

1. 配置web.xml

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <web-app xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xmlns=*"http://xmlns.jcp.org/xml/ns/javaee"*  xsi:schemaLocation=*"http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app\_3\_1.xsd"*  id=*"WebApp\_ID"* version=*"3.1"*>  <display-name>ssm\_dev</display-name>  <welcome-file-list>  <welcome-file>index.html</welcome-file>  <welcome-file>index.htm</welcome-file>  <welcome-file>index.jsp</welcome-file>  <welcome-file>default.html</welcome-file>  <welcome-file>default.htm</welcome-file>  <welcome-file>default.jsp</welcome-file>  </welcome-file-list>  <!-- 加载spring容器 -->  <context-param>  <param-name>contextConfigLocation</param-name>  <param-value>classpath:spring/applicationContext\*.xml</param-value>  </context-param>  <listener>  <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>  </listener>  <!-- 解决post乱码 -->  <filter>  <filter-name>CharacterEncodingFilter</filter-name>  <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>  <init-param>  <param-name>encoding</param-name>  <param-value>utf-8</param-value>  </init-param>  </filter>  <filter-mapping>  <filter-name>CharacterEncodingFilter</filter-name>  <url-pattern>/\*</url-pattern>  </filter-mapping>  <!--定义一个cxf的servlet-->  <servlet>  <servlet-name>CXFServlet</servlet-name>  <servlet-class>org.apache.cxf.transport.servlet.CXFServlet</servlet-class>  </servlet>  <servlet-mapping>  <servlet-name>CXFServlet</servlet-name>  <url-pattern>/webservice/\*</url-pattern>  </servlet-mapping>  </web-app> |

1. 调用服务



## 八．调用接口

1. 服务端和上面的一样就行了
2. 客户端调用

|  |
| --- |
| **package** com.webservice.test;  **import** java.io.BufferedReader;  **import** java.io.DataOutputStream;  **import** java.io.InputStream;  **import** java.io.InputStreamReader;  **import** java.io.StringReader;  **import** java.io.StringWriter;  **import** java.net.HttpURLConnection;  **import** java.net.URL;  **import** java.util.HashMap;  **import** java.util.Map;  **import** freemarker.template.Configuration;  **import** freemarker.template.Template;  **public** **class** HelloWorldTestClient {  **public** **static** **void** main(String[] args) **throws** Exception {  //1.模板加载  String xml = "<soapenv:Envelope xmlns:soapenv=\"http://schemas.xmlsoap.org/soap/envelope/\" xmlns:ser=\"http://services.webservice.tt/\"><soapenv:Header/><soapenv:Body><ser:say><arg0>${line}</arg0></ser:say></soapenv:Body></soapenv:Envelope>";  //其中${line}是map中的key，因为只是一个字符串，所以不用line.name了  Configuration freeMarkerConfig = **null**;  Map<String,Object> map = **new** HashMap<>();  //简单的传一个字符串  String name = "zhangsan";  //传一个对象  //Login login = new Login();  //login.setUsername(name);  //login.setPassword("123");  //map.put("line", login);  //这时xml的模板在渲染时，用${line.username}即可  //传一个list集合时  //List<Login> list = new ArrayList<>();  //list.add(login);  //map.put("loginMdTest", list);  //这时xml的模板在渲染时，用下面例子,注意红色字体  //String xml = "<soapenv:Envelope xmlns:soapenv=\"http://schemas.xmlsoap.org/soap/envelope/\" xmlns:cxf=\"http://cxf.com/\"><soapenv:Header/><soapenv:Body><cxf:get><arg0><loginMdTest><#list loginMdTest as data><loginList><password>${data.password!}</password><username>${data.username!}</username></loginList></#list></loginMdTest></arg0></cxf:get></soapenv:Body></soapenv:Envelope>";  map.put("line", name);    //2.渲染  StringWriter writer = **new** StringWriter(1024);  Template template = **new** Template(name, **new** StringReader(xml), freeMarkerConfig);  template.process(map, writer);  String data = writer.toString();  System.***out***.println("data: "+data);    //3.将数据和模板渲染为SOAP报文，执行http请求  HttpURLConnection connection = **null**;  BufferedReader reader = **null**;  String rs = **null**;  URL url = **new** URL("http://localhost:8080/webService/webservice/hello");    connection = (HttpURLConnection) url.openConnection();  connection.setDoOutput(**true**);  connection.setDoInput(**true**);  connection.setRequestMethod("POST");  connection.setRequestProperty("Content-Type", "text/xml;charset=UTF-8");  connection.setRequestProperty("soapaction", "");  connection.connect();// 连接服务器    DataOutputStream out = **new** DataOutputStream(connection.getOutputStream());  // Java中的char是16位的，一个char存储一个中文字符，直接用writeBytes方法转换会变为8位，直接导致高8位丢失。从而导致中文乱码。  out.write(data.getBytes());  out.close();  StringBuffer sb = **new** StringBuffer();  InputStream is = connection.getInputStream();// 获取数据，真正的请求从此处开始  reader = **new** BufferedReader(**new** InputStreamReader(is, "UTF-8"));  String strRead = **null**;  **while** ((strRead = reader.readLine()) != **null**) {  sb.append(strRead);  }  rs = sb.toString();  System.***out***.println("code: "+connection.getResponseCode());  System.***out***.println("message:　"+connection.getResponseMessage());  System.***out***.println("method: "+connection.getRequestMethod());  System.***out***.println("content-type: "+connection.getRequestProperty("Content-Type"));  System.***out***.println("rs: "+rs);  }  } |

1. 解析xml

|  |
| --- |
| InputStream inputStream = **null**;  // 构造输入流    inputStream = **new** ByteArrayInputStream(rs.getBytes("UTF-8"));    // 解析xml  Document document;  **try** {  document = XmlUtil.*mulThreadParse*(inputStream);  } **catch** (SAXException | IOException | ParserConfigurationException parseXmlE) {  **throw** **new** Exception(parseXmlE.getMessage(), parseXmlE);  }  // 阅读Xml  Element root = document.getDocumentElement();  NodeList flagNodeList = root.getElementsByTagName("ns2:getItemResponse");  **if** (flagNodeList.getLength() == 0) {  *logger*.debug("返回报文异常：无成功标志");  }  Node flagNode = flagNodeList.item(0);  String flag = flagNode.getTextContent();  *logger*.debug("flag: "+flag);  StringBuffer strBuffer = **new** StringBuffer(512);  **if**(flagNodeList != **null** && flagNodeList.getLength()>0){  strBuffer.append(flagNodeList.item(0).getTextContent()+"----").append(";\n");  System.***out***.println("strBuffer: "+strBuffer);  }  **for**(**int** i=0;i<flagNodeList.getLength();i++){  Element el = (Element) flagNodeList.item(i);  NodeList nl = el.getElementsByTagName("barcode");  **for** (**int** j = 0; j < nl.getLength(); j++) {  **if**(nl != **null** && nl.getLength() > 0){  Element ele = (Element) nl.item(j);  String textVal = ele.getFirstChild().getNodeValue();  System.***out***.println("barcode: "+textVal);  }  }  } |