## 1.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：

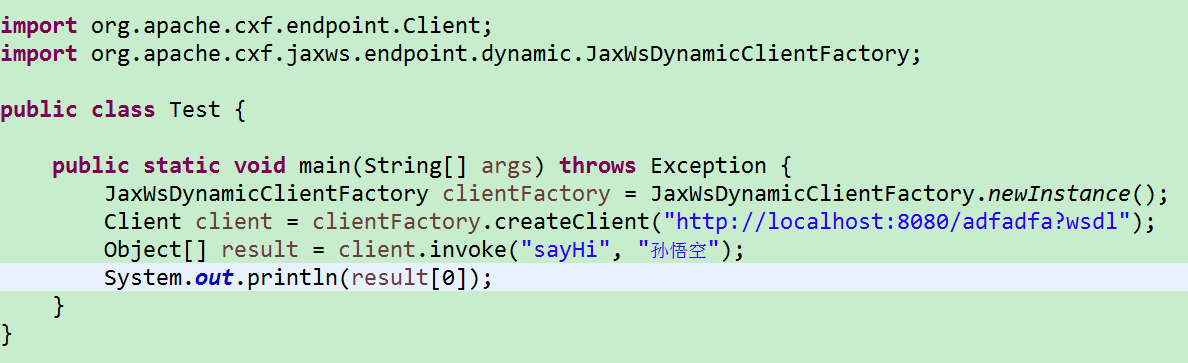


## 2.调用webservice接口

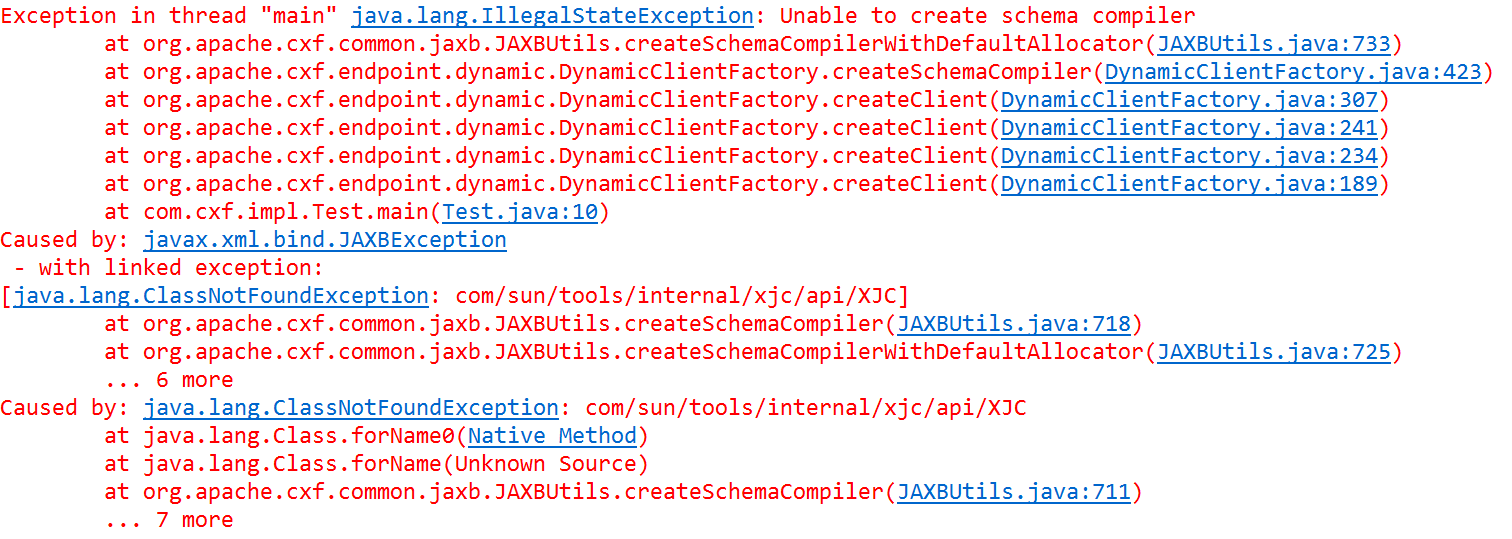
1. 在创建一个web项目
2. 导入cxf的jar包，这里用maven的依赖：

|  |
| --- |
| <!-- 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

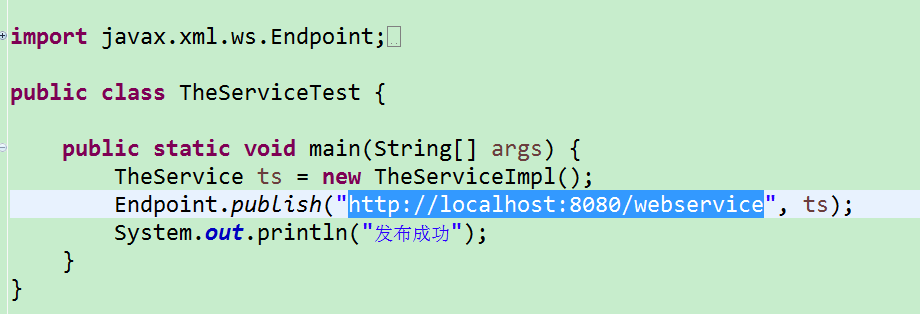
## 添加javabean

1. 服务类

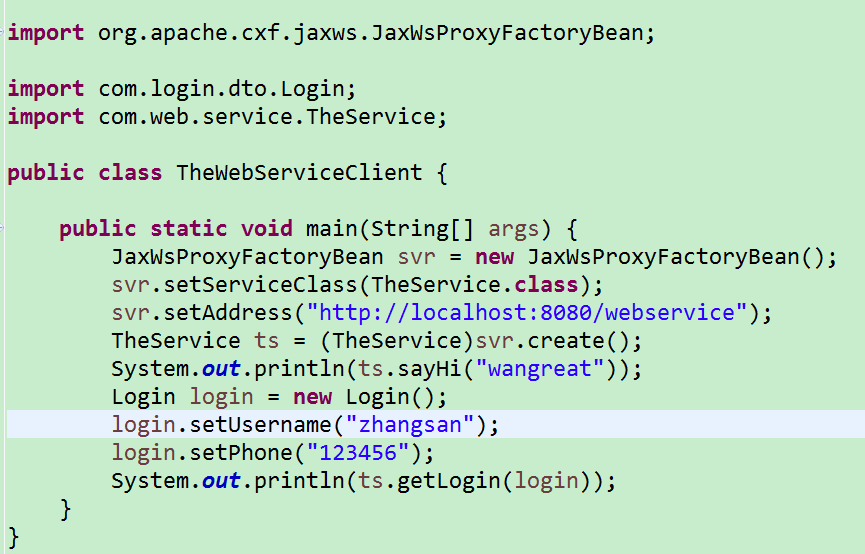




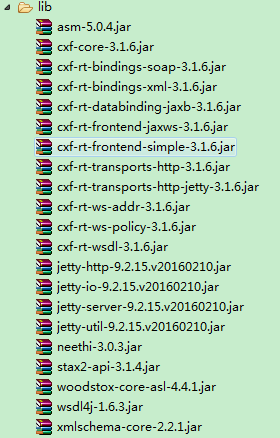
1. 服务发布类



1. 客户端调用类



1. 所用到的jar包：

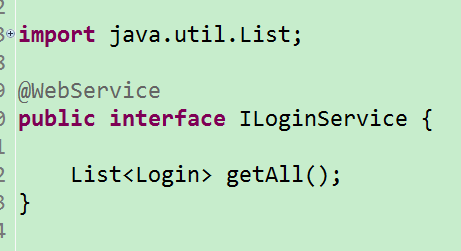


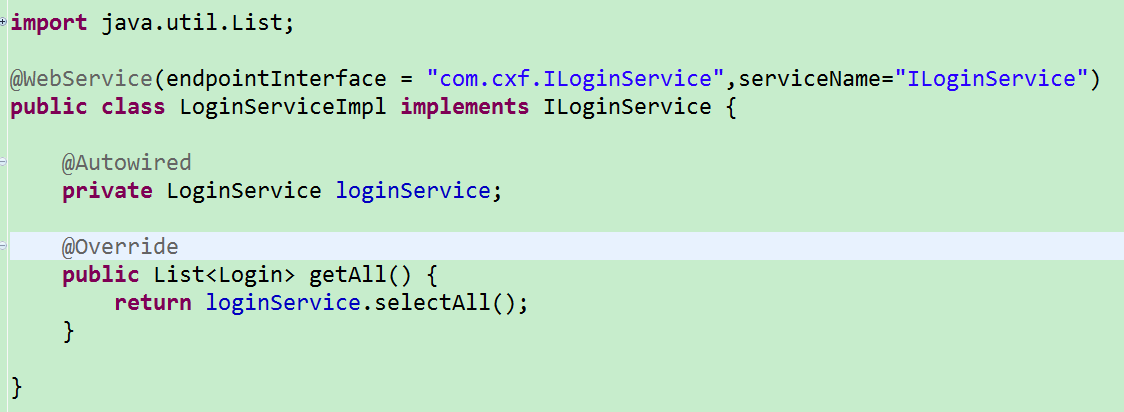
## 4.整合spring

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 --> 80. **<dependency>** 81. **<groupId>**org.apache.cxf**</groupId>** 82. **<artifactId>**cxf-rt-frontend-jaxws**</artifactId>** 83. **<version>**${cxf.version}**</version>** 84. **</dependency>** 85. **<dependency>** 86. **<groupId>**org.apache.cxf**</groupId>** 87. **<artifactId>**cxf-rt-frontend-jaxrs**</artifactId>** 88. **<version>**${cxf.version}**</version>** 89. **</dependency>** 90. **<dependency>** 91. **<groupId>**org.apache.cxf**</groupId>** 92. **<artifactId>**cxf-rt-rs-extension-providers**</artifactId>** 93. **<version>**${cxf.version}**</version>** 94. **</dependency>** 95. **<dependency>** 96. **<groupId>**org.apache.cxf**</groupId>** 97. **<artifactId>**cxf-rt-transports-http**</artifactId>** 98. **<version>**${cxf.version}**</version>** 99. **</dependency>** 100. **<dependency>** 101. **<groupId>**org.codehaus.jettison**</groupId>** 102. **<artifactId>**jettison**</artifactId>** 103. **<version>**1.3.2**</version>** 104. **</dependency>** 105. **<dependency>** 106. **<groupId>**javax.ws.rs**</groupId>** 107. **<artifactId>**jsr311-api**</artifactId>** 108. **<version>**1.1.1**</version>** 109. **</dependency>** 110. <!-- cxf end --> 112. <!-- JSON begin --> 113. **<dependency>** 114. **<groupId>**com.fasterxml.jackson.core**</groupId>** 115. **<artifactId>**jackson-core**</artifactId>** 116. **<version>**${jackson.version}**</version>** 117. **</dependency>** 118. **<dependency>** 119. **<groupId>**com.fasterxml.jackson.core**</groupId>** 120. **<artifactId>**jackson-databind**</artifactId>** 121. **<version>**${jackson.version}**</version>** 122. **</dependency>** 123. **<dependency>** 124. **<groupId>**com.fasterxml.jackson.core**</groupId>** 125. **<artifactId>**jackson-annotations**</artifactId>** 126. **<version>**${jackson.version}**</version>** 127. **</dependency>** 128. **<dependency>** 129. **<groupId>**com.fasterxml.jackson.module**</groupId>** 130. **<artifactId>**jackson-module-jaxb-annotations**</artifactId>** 131. **<version>**${jackson.version}**</version>** 132. **</dependency>** 133. **<dependency>** 134. **<groupId>**com.fasterxml.jackson.jaxrs**</groupId>** 135. **<artifactId>**jackson-jaxrs-json-provider**</artifactId>** 136. **<version>**${jackson.version}**</version>** 137. **</dependency>** 138. **<dependency>** 139. **<groupId>**org.codehaus.jackson**</groupId>** 140. **<artifactId>**jackson-mapper-asl**</artifactId>** 141. **<version>**1.9.0**</version>** 142. **</dependency>** 143. **<dependency>** 144. **<groupId>**org.codehaus.jackson**</groupId>** 145. **<artifactId>**jackson-core-asl**</artifactId>** 146. **<version>**1.9.0**</version>** 147. **</dependency>** 148. <!-- JSON end --> 150. <!-- MyBatis --> 151. **<dependency>** 152. **<groupId>**org.mybatis**</groupId>** 153. **<artifactId>**mybatis**</artifactId>** 154. **<version>**${mybatis.version}**</version>** 155. **</dependency>** 156. **<dependency>** 157. **<groupId>**org.mybatis**</groupId>** 158. **<artifactId>**mybatis-spring**</artifactId>** 159. **<version>**${mybatisspring.version}**</version>** 160. **</dependency>** 161. <!-- AOP begin --> 162. **<dependency>** 163. **<groupId>**org.aspectj**</groupId>** 164. **<artifactId>**aspectjrt**</artifactId>** 165. **<version>**1.7.4**</version>** 166. **</dependency>** 167. **<dependency>** 168. **<groupId>**org.aspectj**</groupId>** 169. **<artifactId>**aspectjweaver**</artifactId>** 170. **<version>**1.7.4**</version>** 171. **</dependency>** 172. **<dependency>** 173. **<groupId>**cglib**</groupId>** 174. **<artifactId>**cglib**</artifactId>** 175. **<version>**3.1**</version>** 176. **</dependency>** 177. <!-- AOP end --> 178. <!-- jdbc driver --> 179. **<dependency>** 180. **<groupId>**localrep.sybase**</groupId>** 181. **<artifactId>**jconn**</artifactId>** 182. **<version>**2.0**</version>** 183. **</dependency>** 184. <!-- connection pool --> 185. **<dependency>** 186. **<groupId>**com.alibaba**</groupId>** 187. **<artifactId>**druid**</artifactId>** 188. **<version>**${druid.version}**</version>** 189. **</dependency>** 190. **</dependencies>** 192. **<build>** 193. **<plugins>** 194. <!-- compiler插件, 设定JDK版本 --> 195. **<plugin>** 196. **<groupId>**org.apache.maven.plugins**</groupId>** 197. **<artifactId>**maven-compiler-plugin**</artifactId>** 198. **<version>**3.0**</version>** 199. **<configuration>** 200. **<source>**${java-version}**</source>** 201. **<target>**${java-version}**</target>** 202. **</configuration>** 203. **</plugin>** 204. **</plugins>** 205. **</build>** 207. <!-- PROGRAMER INFO --> 208. **<organization>** 209. **<name>**Neusoft**</name>** 210. **<url>**http://www.neusoft.com**</url>** 211. **</organization>** 212. **</project>** |

1. 实现服务





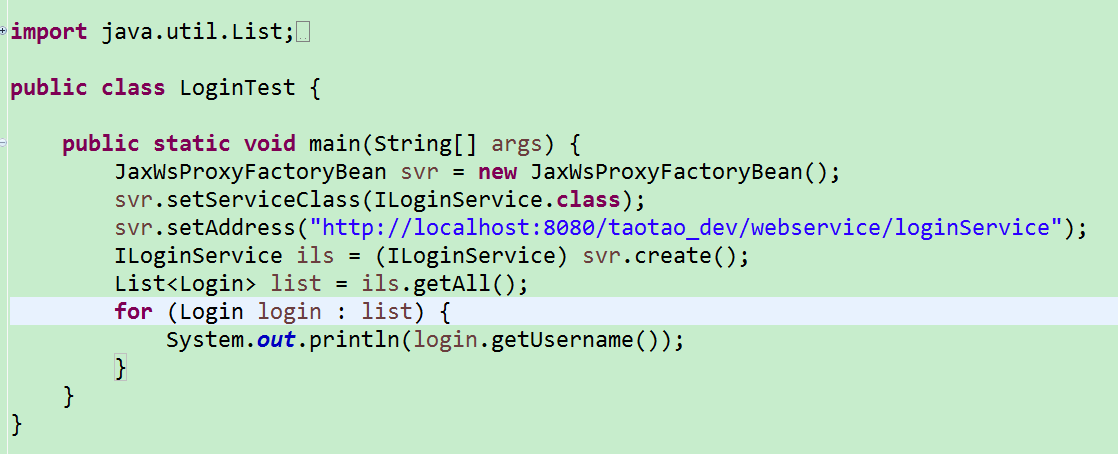
1. 配置applicationContext-cxf.xml

|  |
| --- |
| <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



## 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);  }  } |