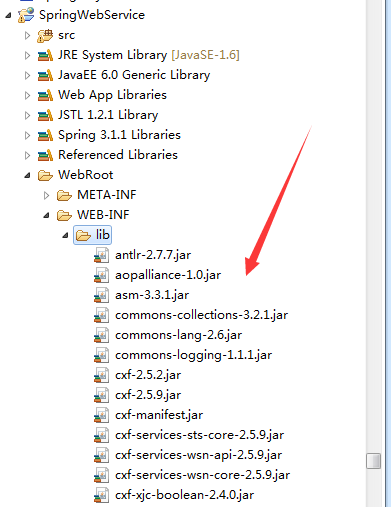
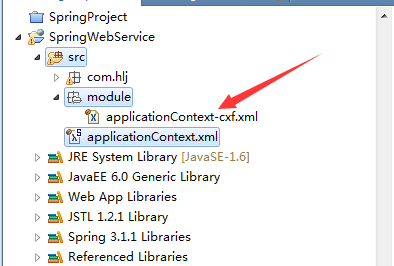
# 1、先建立一个spring项目，再引入cxf 的jar包



# 2、spring配置文件中添加一个关于cxf的配置文件（其实可以写到一起，但是我怕有点乱，就分开写了，这样比较规范）

## 1、spring的配置文件 ApplicationContext.xml 添加cxf的配置文件



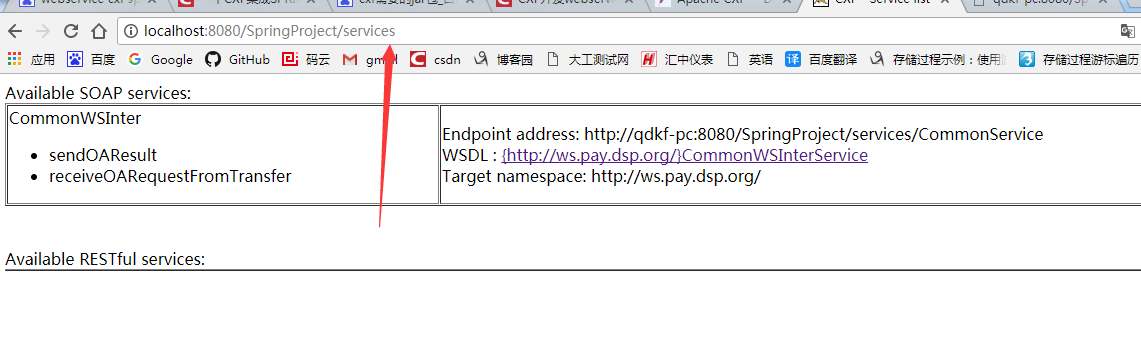
|  |
| --- |
| <context:annotation-config />  <context:component-scan base-package=*"com"* />  <context:component-scan base-package=*"com.hlj.springAnnotationTestSuccess"* />    <import resource=*"classpath:/module/applicationContext-cxf.xml"* />      </beans> |

## 2、先建立一个cxf的配置文件信息（我这里是从中科软复制过来的）

|  |
| --- |
| <?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:soap=*"http://cxf.apache.org/bindings/soap"* xmlns:context=*"http://www.springframework.org/schema/context"*  xmlns:tx=*"http://www.springframework.org/schema/tx"* xmlns:cxf=*"http://cxf.apache.org/core"*  xmlns:jaxrs=*"http://cxf.apache.org/jaxrs"* xmlns:aop=*"http://www.springframework.org/schema/aop"*  xmlns:policy=*"http://cxf.apache.org/policy"* xmlns:wsa=*"http://cxf.apache.org/ws/addressing"*  xsi:schemaLocation=*"*  *http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd*  *http://cxf.apache.org/bindings/soap http://cxf.apache.org/schemas/configuration/soap.xsd*  *http://www.springframework.org/schema/context*  *http://www.springframework.org/schema/context/spring-context-3.0.xsd*  *http://www.springframework.org/schema/aop*  *http://www.springframework.org/schema/aop/spring-aop-3.0.xsd*  *http://cxf.apache.org/core http://cxf.apache.org/schemas/core.xsd*  *http://cxf.apache.org/policy http://cxf.apache.org/schemas/policy.xsd*  *http://cxf.apache.org/jaxws http://cxf.apache.org/schemas/jaxws.xsd*  *http://cxf.apache.org/jaxrs http://cxf.apache.org/schemas/jaxrs.xsd*  *http://www.springframework.org/schema/tx*  *http://www.springframework.org/schema/tx/spring-tx-3.0.xsd*  *"*>  <!-- 引cxf的一些核心配置 ，只要引入了cxf的jar包，这里就有效果了 -->  <import resource=*"classpath:META-INF/cxf/cxf.xml"* />  <import resource=*"classpath:META-INF/cxf/cxf-servlet.xml"* />  <import resource=*"classpath:META-INF/cxf/cxf-extension-soap.xml"* /> |

# 3、web.xml中添加cfx的配置信息 ，service用法看下图

|  |
| --- |
| <!--2、 CXF配置 ， 所有/services请求都会先经过cxf框架-->  <servlet>  <description>Apache CXF EPRK</description>  <display-name>CXFServlet</display-name>  <servlet-name>CXFServlet</servlet-name>  <servlet-class>org.apache.cxf.transport.servlet.CXFServlet</servlet-class>  <load-on-startup>2</load-on-startup>  </servlet>  <servlet-mapping>  <servlet-name>CXFServlet</servlet-name>  <url-pattern>/services/\*</url-pattern>  </servlet-mapping> |



# 4、建立service接口

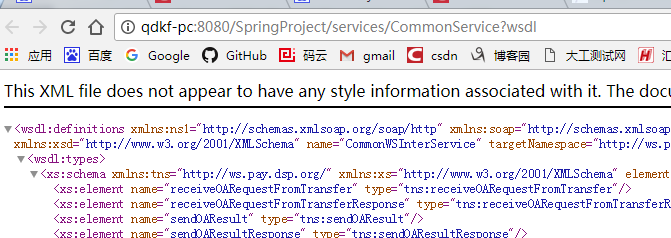
|  |
| --- |
| @WebService(targetNamespace="http://healerjean/")  **public** **interface** CommonWSInter {    **public** String receiveOARequestFromTransfer(@WebParam(name = "name") String name);      String sendOAResult(@WebParam(name = "refNo") String refNo,@WebParam(name = "managerCode") String managerCode );  } |

# 5、添加实现接口

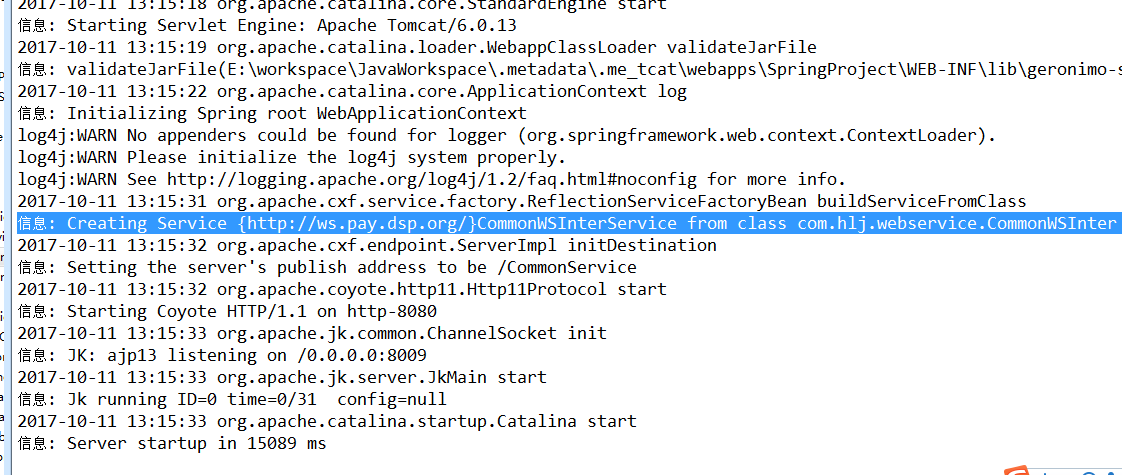
|  |
| --- |
| **public** **class** CommonWSImpl **implements** CommonWSInter{  @Override  **public** String receiveOARequestFromTransfer(String name) {  System.*out*.println("receiveOARequestFromTransfer 方法名字打印");    **return** "receiveOARequestFromTransfer";  }  @Override  **public** String sendOAResult(String refNo, String managerCode) {  // **TODO** Auto-generated method stub  System.*out*.println("sendOAResult 方法名字打印");  **return** "sendOAResult";  } |

# 6、cxf配置spring文件中发布终端

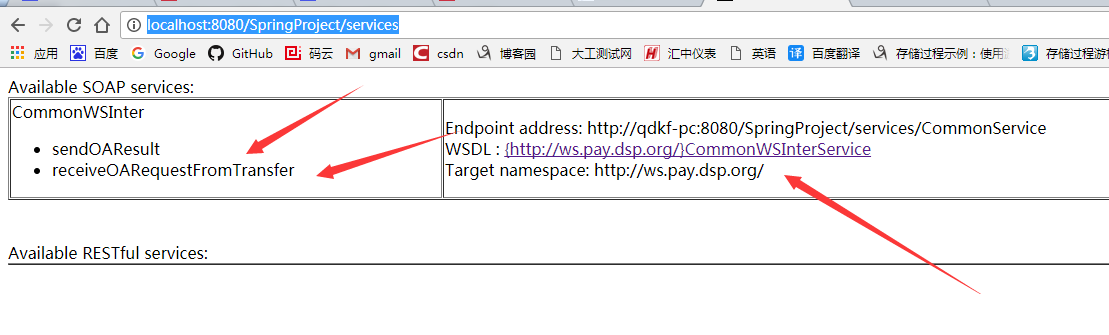
|  |
| --- |
| <!-- 引cxf的一些核心配置 -->  <import resource=*"classpath:META-INF/cxf/cxf.xml"* />  <import resource=*"classpath:META-INF/cxf/cxf-servlet.xml"* />  <import resource=*"classpath:META-INF/cxf/cxf-extension-soap.xml"* />    <!-- 接口 -->  <jaxws:server id=*"commonWS"* serviceClass=*"com.hlj.webservice.CommonWSInter"*  address=*"/CommonService"*> 解释下图  <jaxws:serviceBean>  <ref bean=*"CommonWSImpl"* /> <!-- 引入接口实现类 -->  </jaxws:serviceBean>  </jaxws:server>  <!-- 接口实现类 -->  <bean id=*"CommonWSImpl"* class=*"com.hlj.webservice.CommonWSImpl"* /> |



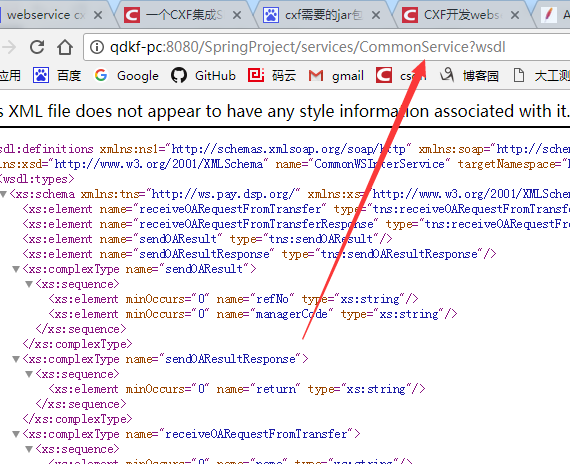
# 7、测试，run运行这个项目



## 1、地址栏中输入<http://localhost:8080/SpringProject/services>



## 2、点击这个链接，打开



# 二、建立客户端，自己调用自己（或者可以通过配置ip模拟调用其他人）

## 1、建立一个客户端的client-beans.xml

|  |
| --- |
| <?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:sec=*"http://cxf.apache.org/configuration/security"*  xmlns:http=*"http://cxf.apache.org/transports/http/configuration"*  xmlns:p=*"http://cxf.apache.org/policy"*  xmlns:jaxws=*"http://java.sun.com/xml/ns/jaxws"*  xmlns:cxf=*"http://cxf.apache.org/core"*  xsi:schemaLocation=*"*  *http://cxf.apache.org/configuration/security*  *http://cxf.apache.org/schemas/configuration/security.xsd*  *http://cxf.apache.org/transports/http/configuration*  *http://cxf.apache.org/schemas/configuration/http-conf.xsd*  *http://cxf.apache.org/core http://cxf.apache.org/schemas/core.xsd*  *http://cxf.apache.org/policy http://cxf.apache.org/schemas/policy.xsd*  *http://www.springframework.org/schema/beans*  *http://www.springframework.org/schema/beans/spring-beans-2.0.xsd"*>    <import resource=*"classpath:META-INF/cxf/cxf.xml"* />  <import resource=*"classpath:META-INF/cxf/cxf-extension-soap.xml"* />  <import resource=*"classpath:META-INF/cxf/cxf-extension-http-binding.xml"*/>  <import resource=*"classpath:META-INF/cxf/cxf-extension-xml.xml"*/>  <import resource=*"classpath:META-INF/cxf/cxf-servlet.xml"* />  <!--  <import resource="classpath:META-INF/cxf/cxf-extension-jaxws.xml" />  -->  <http:conduit name=*"\*.\*"*>  <http:client ConnectionTimeout=*"0"* ReceiveTimeout =*"0"* AutoRedirect=*"true"* Connection=*"Keep-Alive"*/>  </http:conduit>    </beans> |

## 2、实现客户端配置

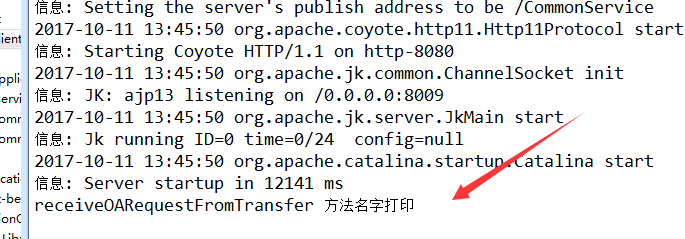
|  |
| --- |
| <bean id=*"commonClientFactory"* class=*"org.apache.cxf.jaxws.JaxWsProxyFactoryBean"*>  <property name=*"serviceClass"* value=*"com.hlj.webservice.CommonWSInter"* />  <property name=*"address"* value=*"http://localhost:8080/SpringProject/services/CommonService"* />  <!--<property name="address" value="http://10.3.181.48:8080/dsp\_oa/services/CommonService" /> -->  </bean>    <bean id=*"commonClient"* class=*"com.hlj.webservice.CommonWSInter"*  factory-bean=*"commonClientFactory"* factory-method=*"create"* /> |

## 3、开始测试

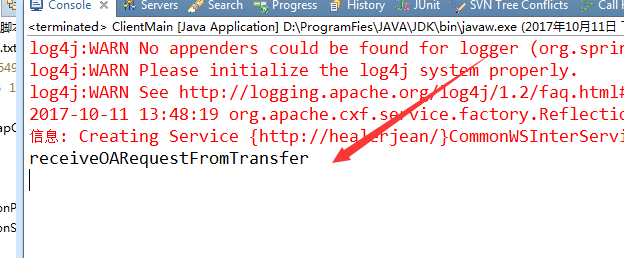
|  |
| --- |
| **public** **class** ClientMain {  **public** **static** **void** main(String[] args) {    ClassPathXmlApplicationContext context = **new** ClassPathXmlApplicationContext("module/client-beans.xml");    CommonWSInter client = (CommonWSInter) context.getBean("commonClient");    String a = client.receiveOARequestFromTransfer("zhang");  System.*out*.println(a);  }  } |

## 4、控制台打印（需要启动tomcat服务）

### 1、tomcat控制台



### 2、main函数控制台



# 三、拦截器拦截报文（入报文和出报文，测试中科软报文日志入库）

## 1、建立入拦截器（对方调用函数时）

|  |
| --- |
| **public** **class** MyInInterceptor **extends** AbstractPhaseInterceptor<Message> {  **private** **static** **final** Logger *log* = Logger.*getLogger*(MyInInterceptor.**class**);    **public** MyInInterceptor() {  //这儿使用receive，接收的意思  **super**(Phase.*RECEIVE*);  }    **public** **void** handleMessage(Message message){    **try** {    InputStream is = message.getContent(InputStream.**class**);    String xml = IOUtils.*toString*(is);    **if**(is != **null**)  message.setContent(InputStream.**class**, is);  } **catch** (Exception e) {  *log*.error("Error when split original inputStream. CausedBy : "+"\n"+e);  } } |

## 2、建立出拦截器，方法调用完成。返回的结果进行拦截

|  |
| --- |
| **public** **class** MyOutInterceptor **extends** AbstractPhaseInterceptor<Message>{  **private** **static** **final** Logger *log* = Logger.*getLogger*(MyOutInterceptor.**class**);    **public** MyOutInterceptor() {  //这儿使用pre\_stream，意思为在流关闭之前  **super**(Phase.*PRE\_STREAM*);  }    **public** **void** handleMessage(Message message) {    **try** {    OutputStream os = message.getContent(OutputStream.**class**);    CachedStream cs = **new** CachedStream();    message.setContent(OutputStream.**class**, cs);    message.getInterceptorChain().doIntercept(message);    CachedOutputStream csnew = (CachedOutputStream) message.getContent(OutputStream.**class**);  InputStream in = csnew.getInputStream();    String xml = IOUtils.*toString*(in);    //这里对xml做处理，处理完后同理，写回流中  IOUtils.*copy*(**new** ByteArrayInputStream(xml.getBytes()), os);    cs.close();  os.flush();    message.setContent(OutputStream.**class**, os);      } **catch** (Exception e) {  *log*.error("Error when split original inputStream. CausedBy : " + "\n" + e);  }  }    **private** **class** CachedStream **extends** CachedOutputStream {    **public** CachedStream() {    **super**();    }    **protected** **void** doFlush() **throws** IOException {    currentStream.flush();    }    **protected** **void** doClose() **throws** IOException {    }    **protected** **void** onWrite() **throws** IOException {    }    } |

## 3、cxf的spring配置文件中添加上述自定义拦截器

|  |
| --- |
| <!-- CXF进入日志拦截器-->  <bean id=*"loggingInInterceptor"* class=*"org.apache.cxf.interceptor.LoggingInInterceptor"* />  <bean id=*"loggingOutInterceptor"* class=*"org.apache.cxf.interceptor.LoggingOutInterceptor"* />  <bean id=*"inMessageInteceptor"* class=*"com.hlj.MyInterceptor.MyInInterceptor"* />  <bean id=*"outMessageInteceptor"* class=*"com.hlj.MyInterceptor.MyOutInterceptor"* />    <cxf:bus>  <cxf:inInterceptors><ref bean=*"inMessageInteceptor"* /></cxf:inInterceptors>  <cxf:outInterceptors><ref bean=*"outMessageInteceptor"* /></cxf:outInterceptors>  </cxf:bus> |

## 4、开始利用map的进行测试

|  |
| --- |
| **public** **class** ClientMain {  **public** **static** **void** main(String[] args) {    ClassPathXmlApplicationContext context = **new** ClassPathXmlApplicationContext("module/client-beans.xml");    CommonWSInter client = (CommonWSInter) context.getBean("commonClient");    Map<String,String> map = **new** HashMap<String,String> ();    map.put("name", "zhang");  map.put("xml", "xml");  map.put("age", "34");    String a = client.receiveOARequestFromTransfer("zhang");  client.sendMap(map);  System.*out*.println(a);  }  } |

## 5、控制台debug测试，观察上面的xml的值

### 1、如拦截器中取得的报文为

|  |
| --- |
| <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"><soap:Body><ns2:sendMap xmlns:ns2="http://healerjean/"><arg0>  <entry>  <key>age</key>  <value>34</value>  </entry>  <entry>  <key>name</key>  <value>zhang</value>  </entry>  <entry>  <key>xml</key>  <value>xml</value>  </entry>  </arg0></ns2:sendMap></soap:Body></soap:Envelope> |

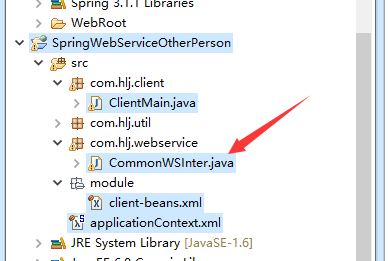
#### 、message为

|  |
| --- |
| {org.apache.cxf.message.Message.PROTOCOL\_HEADERS={Accept=[\*/\*], cache-control=[no-cache], connection=[keep-alive], Content-Length=[323], content-type=[text/xml; charset=UTF-8], host=[localhost:8080], pragma=[no-cache], SOAPAction=[""], user-agent=[Apache CXF 2.5.2]}, HTTP\_CONTEXT\_MATCH\_STRATEGY=stem, http.service.redirection=null, org.apache.cxf.request.url=http://localhost:8080/SpringProject/services/CommonService, org.apache.cxf.request.uri=/SpringProject/services/CommonService, HTTP.REQUEST=org.apache.catalina.connector.RequestFacade@79e59f7c, HTTP.CONFIG=org.apache.catalina.core.StandardWrapperFacade@5249c469, org.apache.cxf.transport.https.CertConstraints=null, Accept=\*/\*, org.apache.cxf.message.Message.PATH\_INFO=/SpringProject/services/CommonService, org.apache.cxf.message.Message.BASE\_PATH=/SpringProject/services/CommonService, org.apache.cxf.message.Message.IN\_INTERCEPTORS=[org.apache.cxf.transport.https.CertConstraintsInterceptor@68fdf3b3], org.apache.cxf.binding.soap.SoapVersion=org.apache.cxf.binding.soap.Soap11@788aa29e, org.apache.cxf.message.Message.ENCODING=UTF-8, org.apache.cxf.message.Message.QUERY\_STRING=null, HTTP.RESPONSE=org.apache.catalina.connector.ResponseFacade@55ba70e5, org.apache.cxf.security.SecurityContext=org.apache.cxf.transport.http.AbstractHTTPDestination$2@3de5da76, org.apache.cxf.request.method=POST, org.apache.cxf.async.post.response.dispatch=true, org.apache.cxf.configuration.security.AuthorizationPolicy=null, org.apache.cxf.message.MessageFIXED\_PARAMETER\_ORDER=false, org.apache.cxf.transport.Destination=org.apache.cxf.transport.servlet.ServletDestination@2cccf2e0, http.base.path=http://localhost:8080/SpringProject, Content-Type=text/xml; charset=UTF-8, HTTP.CONTEXT=org.apache.catalina.core.ApplicationContextFacade@435d671a} |

### 2、出拦截器中取得的报文为 与上面类似

# 6、全局拦截器和自身接口的拦截器（这个名字是我命名的，请观察17.3）

# 四、模拟远程其他用户调用自己，重新建立一个项目，只保留接口类



### 1、测试类不变

|  |
| --- |
| **public** **class** ClientMain {  **public** **static** **void** main(String[] args) {    ClassPathXmlApplicationContext context = **new** ClassPathXmlApplicationContext("module/client-beans.xml");    CommonWSInter client = (CommonWSInter) context.getBean("commonClient");    Map<String,String> map = **new** HashMap<String,String> ();    map.put("name", "zhang");  map.put("xml", "xml");  map.put("age", "34");    String a = client.receiveOARequestFromTransfer("zhang");  client.sendMap(map);  System.*out*.println(a);  } |

### 2、调用成功，另外的项目服务端有反应