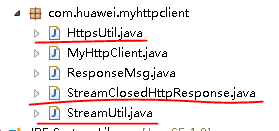
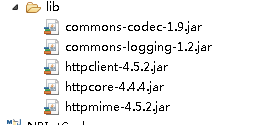
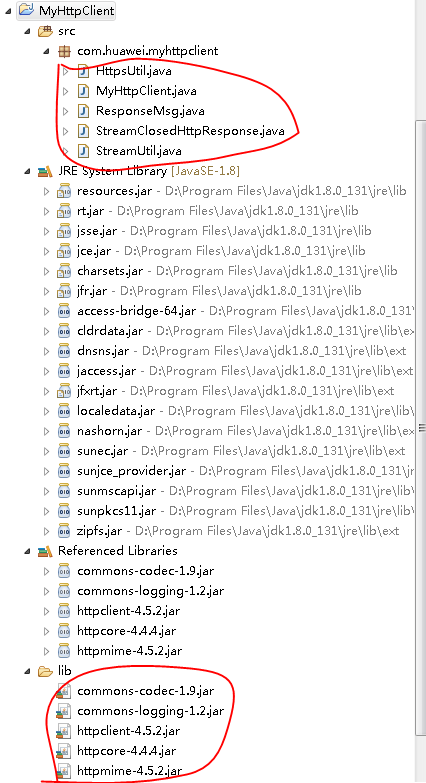
使用C#调用北向接口说明

当你使用C#调用北向接口时，由于需要https的双向认证，而C#框架只提供了一个设置证书的接口。  
此时可以将北向liteDemo的https请求部分打成jar包后转成dll供c#工程使用。  
 具体步骤如下：  
先创建一个java工程，将liteDemo源码中的如下文件导入工程  
  
MyHttpClient.java文件代码

package com.huawei.myhttpclient;  
import java.util.Map;  
  
  
public class MyHttpClient   
{          
        protected static String SELFCERTPATH = "";  
        protected static String TRUSTCAPATH = "";  
  
        protected static String SELFCERTPWD = "";  
        protected static String TRUSTCAPWD = "";  
  
        public HttpsUtil httpsUtil;  
  
        private static MyHttpClient instance;  
  
        private MyHttpClient(String SELFCERTPATH, String SELFCERTPWD,   
                        String TRUSTCAPATH, String TRUSTCAPWD)  
        {  
                MyHttpClient.SELFCERTPATH = SELFCERTPATH;  
                MyHttpClient.SELFCERTPWD = SELFCERTPWD;  
                MyHttpClient.TRUSTCAPATH = TRUSTCAPATH;  
                MyHttpClient.TRUSTCAPWD = TRUSTCAPWD;  
  
                httpsUtil = new HttpsUtil();  
  
        try   
        {  
                        httpsUtil.initSSLConfigForTwoWay();  
                }  
        catch (Exception e1)  
        {  
                        e1.printStackTrace();  
                }  
        }  
  
        public static MyHttpClient getInstance(String SELFCERTPATH, String SELFCERTPWD,  
                        String TRUSTCAPATH, String TRUSTCAPWD)  
        {  
                if(null == instance)  
                {  
                        instance = new MyHttpClient(SELFCERTPATH, SELFCERTPWD,  
                                        TRUSTCAPATH, TRUSTCAPWD);  
                }  
                return instance;  
        }  
  
        public ResponseMsg sendMsg(String reqUrl, Map<String, String> param)  
        {  
                ResponseMsg result = new ResponseMsg();  
  
        StreamClosedHttpResponse responseLogin = null;  
                try   
                {  
                        responseLogin = httpsUtil.doPostFormUrlEncodedGetStatusLine(reqUrl, param);  
                }   
                catch (Exception e)   
                {  
                        e.printStackTrace();  
                }  
  
                if(null != responseLogin)  
                {  
                        if(null != responseLogin.getStatusLine())  
                        {  
result.setStatusCode(responseLogin.getStatusLine().getStatusCode());  
                                result.setContent(responseLogin.getContent());  
                        }  
                }  
  
        return result;  
        }  
  
        @SuppressWarnings("deprecation")  
        public void closeConn()  
        {  
                this.httpsUtil.close();  
                instance = null;  
        }

ResponseMsg.java代码如下

package com.huawei.myhttpclient;  
  
public class ResponseMsg {  
        private int statusCode = -1;  
        private String content;  
  
        public int getStatusCode() {  
                return statusCode;  
        }  
  
        public void setStatusCode(int statusCode) {  
                this.statusCode = statusCode;  
        }  
  
        public String getContent() {  
                return content;  
        }  
  
        public void setContent(String content) {  
                this.content = content;  
        }  
  
}

导入必须的依赖包  
  
  
最终工程目录如下：  
  
  
生成了jar包后 ，接下来就是将jar包转成dll了 

jar包转成dll

IKVM.NET是一个针对Mono和微软.net框架的java实现，其设计目的是在.NET平台上运行java程序。

有些在java程序上可以实现的功能，但在C#中无法实现或者目前不知道怎么实现时（如带有特殊Padding的RSA2048加密，https双向认证等），可以考虑使用IKVM工具把已生成的jar转成dll供C#程序去调用。

IKVM官网地址：https://www.nuget.org/packages/IKVM/

由于IKVM7最高只支持jdk1.6编译出来的jar转换，建议使用最新版本IKVM8。

**使用说明**

前提条件：已通过java实现所需功能，且编译成一个jar，如MyHttpClient.jar。

1. 在官网下载ikvm压缩包，这里用的是ikvmbin-8.1.5717.0.zip，解压到一个文件夹中，如C:\tools\ikvm-8.1.5717.0。

2. 配置环境变量：在系统环境变量中增加IKVM\_HOME：C:\tools\ikvm-8.1.5717.0，同时Path中增加%IKVM\_HOME%/bin;

3. 把MyHttpClient.jar放到一个工作目录下，如C:\tools\ikvm，在命令行中进入该目录，执行 ikvmc -target:library MyHttpClient.jar，即在当前目录下生产MyHttpClient.dll，这里只需要生成一个dll，如需生产可执行程序，可执行命令 ikvmc -target:exe xxx.jar。

4. 在C#工程中引用MyHttpClient.dll以及IKVM相关的dll（C:\tools\ikvm-8.1.5717.0\bin目录下的以IKVM.开头的dll，这里为了方便全部引用了），C#中类的命名空间和java中的类的包名是一致的，如java中import com.huawei.myhttpclient，C#中为using com.huawei.myhttpclient

**特别注意：**

由于实际的java实现工程很多情况下都依赖一些第三方jar，比如举例的MyHttpClient工程使用了Apache httpClient相关组件实现的设置客户端，服务端证书，发送消息功能，依赖的jar有：commons-codec-1.9.jar，commons-logging-1.2.jar，httpclient-4.5.2.jar，httpcore-4.4.4.jar，httpmime-4.5.2.jar，在网上查询很多做法是把这些第三方jar依次编译成dll，执行ikvm命令时需要指定指定依赖的dll，比较麻烦，这里的做法是把这些jar用winrar等解压软件打开，然后里面的根目录文件夹org（里面包含了Apache httpClient编译生成的class）,拷到MyHttpClient.jar中。