链滴

求一个 httpClient 4.5.x 工具类

作者：manyue
原文链接：https://ld246.com/article/1493868446952
来源网站：链滴
许可协议：署名-相同方式共享 4.0 国际 (CC BY-SA 4.0)

求一个httpClient 4.5.x的工具类，求大佬们赏赐一个！

希望包含上传文件和下载文件的功能，能支持HTTPS

不做和伸手党，会给分的 blush

下面是按照百度写的代码，感觉有点糟心，提点建议也谢谢了~

```
package com.fadada.sdk.util.http;

import java.io.IOException;
import java.security.KeyManagementException;
import java.security.NoSuchAlgorithmException;
import java.security.cert.CertificateException;
import java.security.cert.X509Certificate;
import java.util.Arrays;
import java.util.List;

import javax.net.ssl.SSLContext;
import javax.net.ssl.TrustManager;
import javax.net.ssl.X509TrustManager;

import org.apache.http.Consts;
import org.apache.http.HttpEntity;
import org.apache.http.HttpStatus;
import org.apache.http.NameValuePair;
import org.apache.http.client.config.AuthSchemes;
import org.apache.http.client.config.CookieSpecs;
import org.apache.http.client.config.RequestConfig;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.CloseableHttpResponse;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.config.Registry;
import org.apache.http.config.RegistryBuilder;
import org.apache.http.conn.socket.ConnectionSocketFactory;
import org.apache.http.conn.socket.PlainConnectionSocketFactory;
import org.apache.http.conn.ssl.NoopHostnameVerifier;
import org.apache.http.conn.ssl.SSLConnectionSocketFactory;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.impl.conn.PoolingHttpClientConnectionManager;
import org.apache.http.util.EntityUtils;

public class Zhou {

    public static CloseableHttpResponse doGet(String url, String cookie, String refer) throws IO
xception {
        // http实现get请求:首先设置全局的标准cookie策略
        RequestConfig config = RequestConfig.custom().setCookieSpec(CookieSpecs.STANDAR
_STRICT).build();
        // 设置可关闭的httpclient
```

```
        CloseableHttpClient httpClient = HttpClients.custom().setDefaultRequestConfig(config).
uild();
        // 发送get请求
        HttpGet get = new HttpGet(url);
        if (cookie != null) {
            get.setHeader("Cookie", cookie);
        }
        if (refer != null) {
            get.setHeader("Refer", refer);
        }
        CloseableHttpResponse response = httpClient.execute(get);
        return response;
    }

    public static CloseableHttpResponse doPost(String url, List<NameValuePair> values, String
cookie, String refer) throws IOException {
        // http实现post请求:首先设置全局的标准cookie策略
        RequestConfig config = RequestConfig.custom().setCookieSpec(CookieSpecs.STANDAR
_STRICT).build();
        // 设置可关闭的httpclient
        CloseableHttpClient httpClient = HttpClients.custom().setDefaultRequestConfig(config).
uild();
        // 对请求参数进行编码后再进行发送
        HttpPost post = new HttpPost(url);
        if (cookie != null) {
            post.setHeader("Cookie", cookie);
        }
        if (refer != null) {
            post.setHeader("Refer", refer);
        }
        UrlEncodedFormEntity entity = new UrlEncodedFormEntity(values, Consts.UTF_8);
        post.setEntity(entity);
        CloseableHttpResponse response = httpClient.execute(post);
        return response;
    }

    private static TrustManager manager = new X509TrustManager() {

        @Override
        public X509Certificate[] getAcceptedIssuers() {
            return null;
        }

        @Override
        public void checkServerTrusted(X509Certificate[] arg0, String arg1) throws CertificateExc
ption {

        }

        @Override
        public void checkClientTrusted(X509Certificate[] arg0, String arg1) throws CertificateExc
ption {

        }
```

```
    };

    private static SSLConnectionSocketFactory socketFactory;// 私密连接工厂

    private static void enableSSL() {
        // https网站一般情况下使用了安全系数较低的SHA-1签名，因此首先我们在调用SSL之前需要
写验证方法，取消检测SSL。
        try {
            SSLContext context = SSLContext.getInstance("TLS");
            context.init(null, new TrustManager[] { manager }, null);
            socketFactory = new SSLConnectionSocketFactory(context, NoopHostnameVerifier.IN
TANCE);
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        } catch (KeyManagementException e) {
            e.printStackTrace();
        }
    }

    public static CloseableHttpResponse doHttpsGet(String url, String cookie, String refer) thr
ws IOException {
        enableSSL();
        RequestConfig defaultRequestConfig = RequestConfig.custom().setCookieSpec(CookieS
ecs.STANDARD_STRICT).setExpectContinueEnabled(true).setTargetPreferredAuthSchemes(Arr
ys.asList(AuthSchemes.NTLM, AuthSchemes.DIGEST)).setProxyPreferredAuthSchemes(Arrays.a
List(AuthSchemes.NTLM)).build();
        Registry<ConnectionSocketFactory> socketFactoryRegistry = RegistryBuilder.<Connecti
nSocketFactory> create().register("http", PlainConnectionSocketFactory.INSTANCE).register("h
tps", socketFactory).build();
        // 创建ConnectionManager，添加Connection配置信息
        PoolingHttpClientConnectionManager connectionManager = new PoolingHttpClientCon
ectionManager(socketFactoryRegistry);
        CloseableHttpClient httpClient = HttpClients.custom().setConnectionManager(connectio
Manager).setDefaultRequestConfig(defaultRequestConfig).build();
        // 发送get请求
        HttpGet get = new HttpGet(url);
        if (cookie != null) {
            get.setHeader("Cookie", cookie);
        }
        if (refer != null) {
            get.setHeader("Refer", refer);
        }
        CloseableHttpResponse response = httpClient.execute(get);
        return response;
    }

    public static CloseableHttpResponse doHttpsPost(String url, List<NameValuePair> values, S
ring cookie, String refer) throws IOException {
        enableSSL();
        RequestConfig defaultRequestConfig = RequestConfig.custom().setCookieSpec(CookieS
ecs.STANDARD_STRICT).setExpectContinueEnabled(true).setTargetPreferredAuthSchemes(Arr
ys.asList(AuthSchemes.NTLM, AuthSchemes.DIGEST)).setProxyPreferredAuthSchemes(Arrays.a
List(AuthSchemes.NTLM)).build();
        Registry<ConnectionSocketFactory> socketFactoryRegistry = RegistryBuilder.<Connecti
```

```
nSocketFactory> create().register("http", PlainConnectionSocketFactory.INSTANCE).register("h
tps", socketFactory).build();
        // 创建ConnectionManager，添加Connection配置信息
        PoolingHttpClientConnectionManager connectionManager = new PoolingHttpClientCon
ectionManager(socketFactoryRegistry);
        CloseableHttpClient httpClient = HttpClients.custom().setConnectionManager(connectio
Manager).setDefaultRequestConfig(defaultRequestConfig).build();
        // 对请求参数进行编码后再进行发送
        HttpPost post = new HttpPost(url);
        if (cookie != null) {
            post.setHeader("Cookie", cookie);
        }
        if (refer != null) {
            post.setHeader("Refer", refer);
        }
        UrlEncodedFormEntity entity = new UrlEncodedFormEntity(values, Consts.UTF_8);
        post.setEntity(entity);
        CloseableHttpResponse response = httpClient.execute(post);
        return response;
    }

    public static String doHttpsPost(String url, HttpEntity entity) {
        CloseableHttpClient httpClient = null;
        CloseableHttpResponse response = null;
        try {
            enableSSL();
            RequestConfig defaultRequestConfig = RequestConfig.custom().setCookieSpec(Cooki
Specs.STANDARD_STRICT).setExpectContinueEnabled(true).setTargetPreferredAuthSchemes(A
rays.asList(AuthSchemes.NTLM, AuthSchemes.DIGEST)).setProxyPreferredAuthSchemes(Arrays
asList(AuthSchemes.NTLM)).build();
            Registry<ConnectionSocketFactory> socketFactoryRegistry = RegistryBuilder.<Connec
ionSocketFactory> create().register("http", PlainConnectionSocketFactory.INSTANCE).register(
https", socketFactory).build();
            // 创建ConnectionManager，添加Connection配置信息
            PoolingHttpClientConnectionManager connectionManager = new PoolingHttpClientC
nnectionManager(socketFactoryRegistry);
            httpClient = HttpClients.custom().setConnectionManager(connectionManager).setDef
ultRequestConfig(defaultRequestConfig).build();
            // 对请求参数进行编码后再进行发送
            HttpPost post = new HttpPost(url);
            post.setEntity(entity);
            response = httpClient.execute(post);
            if (response.getStatusLine().getStatusCode() == HttpStatus.SC_OK) {
                HttpEntity respEntity = response.getEntity();
                if (null == entity) {
                    throw new RuntimeException("HttpEntity is null.");
                }
                return EntityUtils.toString(respEntity, Consts.UTF_8);
            } else {
                throw new RuntimeException("connect fail. http_status:" + response.getStatusLine()
getStatusCode());
            }
        } catch (Exception e) {
            e.printStackTrace();
```

```
            throw new RuntimeException(e);
        }finally{
            try {
                // 关闭连接,释放资源
                if (response != null) {
                    response.close();
                }
                if (httpClient != null) {
                    httpClient.close();
                }
            } catch (IOException e) {
                e.printStackTrace();
            }
        }

    }

}
```