HttpClient 用于 https 交互

作者：linyiheng
原文链接：https://ld246.com/article/1503242895272
来源网站：链滴

httpclient 4.1.2

HttpClient用于https交互

接受任意ssl证书的HttpClient

```
    @SuppressWarnings("finally")
    public static String sendhttpsRequest(String requestUrl,Map<String, String> requestHeade
s, String requestBody) {
        //boolean isSuccess = false;
        long responseLength = 0; // 响应长度
        String responseContent = null; // 响应内容
        StringEntity strEntity = null;
        HttpClient httpClient=new DefaultHttpClient();
        try {
            strEntity = new StringEntity(requestBody, HTTP.UTF_8);
            SSLContext ctx = SSLContext.getInstance("TLS");
            X509TrustManager tm = new X509TrustManager() {
                public X509Certificate[] getAcceptedIssuers() {
                    return null;
                }

                public void checkClientTrusted(X509Certificate[] arg0,
                        String arg1) throws CertificateException {
                }

                public void checkServerTrusted(X509Certificate[] arg0,
                        String arg1) throws CertificateException {
                }
            };
            ctx.init(null, new TrustManager[] { tm }, null);
            //接收任意证书的设置
            SSLSocketFactory ssf = new SSLSocketFactory(ctx,SSLSocketFactory.ALLOW_ALL_HOS
NAME_VERIFIER);
            httpClient.getConnectionManager().getSchemeRegistry()
                    .register(new Scheme("https", 443, ssf));
            HttpPost httpPost = new HttpPost(requestUrl);
            httpPost.setEntity(strEntity);
            Iterator<?> it = requestHeaders.entrySet().iterator();
            while (it.hasNext()) {
                Map.Entry<String, String> entry = (Map.Entry<String, String>) it.next();
                httpPost.setHeader(entry.getKey(), entry.getValue());
            }
            HttpResponse response;
            HttpEntity entity;
            response = httpClient.execute((HttpUriRequest) httpPost);
            entity = response.getEntity(); // 获取响应实体
            if (null != entity) {
                responseLength = entity.getContentLength();
                responseContent = EntityUtils.toString(entity, "UTF-8");
                EntityUtils.consume(entity); // Consume response content
            }
```

原文链接：HttpClient 用于 https 交互

```
            System.out.println("请求地址: " + httpPost.getURI());
            System.out.println("响应状态: " + response.getStatusLine());
            System.out.println("响应长度: " + responseLength);
            System.out.println("响应内容: " + responseContent);
        } catch (UnsupportedEncodingException e) {
            e.printStackTrace();
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        } catch (KeyManagementException e) {
            e.printStackTrace();
        } catch (ClientProtocolException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }finally {
            httpClient.getConnectionManager().shutdown(); // 关闭连接,释放资源
            return responseContent;
        }


    }
```

接受指定服务端ssl证书的HttpClient

```
    @SuppressWarnings("finally")
    public static boolean sendhttpsRequest(String requestUrl,String key, String message) {
        boolean bool = false;
        long responseLength = 0; // 响应长度
        String responseContent = null; // 响应内容
        HttpClient httpClient = new DefaultHttpClient(); // 创建默认的httpClient实例
        int port=getPortFromUrl(requestUrl);
        try {
            FileInputStream fis=null;
            KeyStore trustStore = KeyStore.getInstance("pkcs12");
    //            FileInputStream fis = new FileInputStream(new File("D:/workspace/mdm/MDMSer
er/etc/server.p12"));
                //用户测试
            fis = new FileInputStream(new File("D:/workspace/mdm/MDMServer/etc/server.p12"));
            try {
                trustStore.load(fis, "1111".toCharArray()); // 加载KeyStore
            } catch (NoSuchAlgorithmException e) {
                e.printStackTrace();
            } catch (CertificateException e) {
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            } finally {
                try {
                    fis.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
            SSLSocketFactory socketFactory = new SSLSocketFactory(trustStore); // 创建Socket工厂
```

将trustStore注入

```
        Scheme sch = new Scheme("https", port, socketFactory); // 创建Scheme
        httpClient.getConnectionManager().getSchemeRegistry().register(sch); // 注册Scheme
        HttpPost httpPost = new HttpPost(requestUrl); // 创建HttpGet

        List<NameValuePair> params = new ArrayList<NameValuePair>();
        Map map = new HashMap();
        map.put(key, message);
        Iterator<?> it = map.entrySet().iterator();
        while (it.hasNext()) {
            Map.Entry entry = (Map.Entry<?, ?>) it.next();
            BasicNameValuePair pa = new BasicNameValuePair(entry.getKey()
                    + "", entry.getValue() + "");
            params.add(pa);
        }
        // 封包添加到Post请求
        try {
            httpPost.setEntity(new UrlEncodedFormEntity(params, HTTP.UTF_8));
        } catch (UnsupportedEncodingException e1) {
            e1.printStackTrace();
        }

        // httpPost.setRequestBody(message);
        HttpResponse response = httpClient
                .execute((HttpUriRequest) httpPost);// 执行GET请求
        HttpEntity entity = response.getEntity(); // 获取响应实体
        if (null != entity) {
            responseLength = entity.getContentLength();
            responseContent = EntityUtils.toString(entity, "UTF-8");
            EntityUtils.consume(entity); // Consume response content
        }
        System.out.println("请求地址: " + httpPost.getURI());
        System.out.println("响应状态: " + response.getStatusLine());
        System.out.println("响应长度: " + responseLength);
        System.out.println("响应内容: " + responseContent);
        bool = true;
    } catch (KeyManagementException e) {
        e.printStackTrace();
    } catch (UnrecoverableKeyException e) {
        e.printStackTrace();
    } catch (KeyStoreException e) {
        e.printStackTrace();
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (NoSuchAlgorithmException e) {
        e.printStackTrace();
    } catch (ClientProtocolException e) {
        e.printStackTrace();
    } catch (ParseException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        httpClient.getConnectionManager().shutdown(); // 关闭连接,释放资源
```

原文链接：HttpClient 用于 https 交互

```
            return bool;
        }
    }
```

HttpClient4.4版

```
    public static CloseableHttpClient createSSLClientDefault() {
        try {
            SSLContext sslContext = new SSLContextBuilder().loadTrustMaterial(
                    null, new TrustStrategy() {
                        public boolean isTrusted(X509Certificate[] chain,
                                String authType) throws CertificateException {
                            return true;
                        }
                    }).build();
            SSLConnectionSocketFactory sslsf = new SSLConnectionSocketFactory(
                    sslContext);
            return HttpClients.custom().setSSLSocketFactory(sslsf).build();
        } catch (Exception e) {
            e.printStackTrace();
        }
        return HttpClients.createDefault();
    }

    public static OAuthToken sendOAuthTokenPostRequest(String url,String message) {
        StringEntity  entity=new StringEntity(message,ContentType.create("application/x-www-f
rm-urlencoded",Consts.UTF_8));
        System.out.println(entity.getContentLength());
        String oAuthTokenStr="";
        CloseableHttpClient httpClient = ClientTool.createSSLClientDefault();
        HttpPost post=new HttpPost();
        try {
            post.setURI(new URI(url));
            post.setEntity(entity);
            HttpResponse response = httpClient.execute(post);
            HttpEntity respEntity = response.getEntity();
            List<String> ls = null;
            if (respEntity != null) {
                respEntity = new BufferedHttpEntity(respEntity);
                InputStream in = respEntity.getContent();
                ls = IOUtils.readLines(in, "UTF-8");
                in.close();
            }
            for (String e : ls) {
                oAuthTokenStr+=e;
            }

        } catch (Exception e) {
            e.printStackTrace();
        }
        System.out.println(oAuthTokenStr);
        JSONObject oAuthToken=JSONObject.fromObject(oAuthTokenStr);
        OAuthToken oat=(OAuthToken) JSONObject.toBean(oAuthToken, OAuthToken.class);
        return oat;
```

```
}
```