14、java发起http请求

# 1、post请求

## 1、http工具类

public class HttpHelper {  
  
 public static final String ***ENCODING\_UTF8*** = "UTF-8";  
 public static final String ***ENCODING\_GBK*** = "GBK";  
  
 public static final String ***POST*** = "POST";  
 public static final String ***GET*** = "GET";  
  
  
 private static RequestConfig defaultRequestConfig() {  
 return RequestConfig.custom().setSocketTimeout(1500).setConnectTimeout(1500).setConnectionRequestTimeout(1500).build();  
 }  
  
 public static String handleGet(String uri) {  
 CloseableHttpClient httpclient = HttpClients.custom().build();  
 try {  
 HttpGet request = new HttpGet(uri);  
 request.setHeader("User-Agent", "admore");  
 request.setConfig(defaultRequestConfig());  
 HttpResponse response = httpclient.execute(request);  
 return IOUtils.toString(response.getEntity().getContent(), ***ENCODING\_UTF8***);  
 } catch (IOException e) {  
 throw new RuntimeException(e);  
 } finally {  
 IOUtils.closeQuietly(httpclient);  
 }  
 }  
  
 public static String handleGetHttps(String uri) {  
 CloseableHttpClient httpclient = null;  
 try {  
 SSLContext sslcontext = SSLContexts.custom().loadTrustMaterial(new TrustSelfSignedStrategy()).build();  
 sslcontext.init(null, new TrustManager[]{new MyTrustManager()}, null);  
  
 httpclient = HttpClients.custom().setSSLSocketFactory(new SSLConnectionSocketFactory(sslcontext)).build();  
 HttpGet request = new HttpGet(uri);  
 request.setHeader("User-Agent", "admore");  
 request.setConfig(defaultRequestConfig());  
 HttpResponse response = httpclient.execute(request);  
 return IOUtils.toString(response.getEntity().getContent(), ***ENCODING\_UTF8***);  
 } catch (Exception e) {  
 throw new RuntimeException(e);  
 } finally {  
 IOUtils.closeQuietly(httpclient);  
 }  
 }  
  
 public static String handlePost(String url, Map<String, String> formParams) {  
 CloseableHttpClient httpclient = HttpClients.custom().build();  
 try {  
 HttpPost request = new HttpPost(url);  
 List<NameValuePair> formValue = new ArrayList<>();  
 for (Map.Entry<String, String> entry : formParams.entrySet()) {  
 formValue.add(new BasicNameValuePair(entry.getKey(), entry.getValue()));  
 }  
 request.setEntity(new UrlEncodedFormEntity(formValue, "UTF-8"));  
 request.setHeader("User-Agent", "admore");  
 request.setConfig(defaultRequestConfig());  
 HttpResponse response = httpclient.execute(request);  
 return IOUtils.toString(response.getEntity().getContent(), ***ENCODING\_UTF8***);  
 } catch (IOException e) {  
 throw new RuntimeException(e);  
 } finally {  
 IOUtils.closeQuietly(httpclient);  
 }  
 }  
  
 public static String handlePost(String url, Map<String, String> formParams, Integer timout) {  
 CloseableHttpClient httpclient = HttpClients.custom().build();  
 try {  
 HttpPost request = new HttpPost(url);  
 List<NameValuePair> formValue = new ArrayList<>();  
 for (Map.Entry<String, String> entry : formParams.entrySet()) {  
 formValue.add(new BasicNameValuePair(entry.getKey(), entry.getValue()));  
 }  
 request.setEntity(new UrlEncodedFormEntity(formValue, "UTF-8"));  
 request.setHeader("User-Agent", "admore");  
 RequestConfig config = RequestConfig.custom()  
 .setConnectTimeout(timout) //连接超时  
 .setSocketTimeout(timout) //读超时  
 .build();  
 request.setConfig(config);  
 // request.setConfig(defaultRequestConfig());  
 HttpResponse response = httpclient.execute(request);  
 return IOUtils.toString(response.getEntity().getContent(), ***ENCODING\_UTF8***);  
 } catch (IOException e) {  
 throw new RuntimeException(e);  
 } finally {  
 IOUtils.closeQuietly(httpclient);  
 }  
 }  
  
 public static String handlePost(String url, String content) {  
 return handlePost(url, content, ***ENCODING\_UTF8***);  
 }  
  
 public static String handlePost(String url, String content, String encoding) {  
 InputStream input = null;  
 OutputStream output = null;  
 HttpURLConnection connection = null;  
 content = StringUtils.defaultIfEmpty(content, "");  
  
 try {  
 URL request = new URL(url);  
 connection = (HttpURLConnection) request.openConnection();  
 connection.setRequestMethod(***POST***);  
 connection.setDoOutput(true);  
 output = connection.getOutputStream();  
 output.write(content.getBytes(encoding));  
 input = connection.getInputStream();  
 return IOUtils.toString(input, encoding);  
 } catch (IOException e) {  
 throw new RuntimeException(e);  
 } finally {  
 IOUtils.closeQuietly(input);  
 IOUtils.closeQuietly(output);  
 if (connection != null) {  
 connection.disconnect();  
 }  
 }  
 }  
  
  
  
 public static class MyTrustManager extends X509ExtendedTrustManager {  
  
 public void checkClientTrusted(X509Certificate[] chain, String authType) throws CertificateException {  
 }  
  
 public void checkServerTrusted(X509Certificate[] chain, String authType) throws CertificateException {  
 }  
  
 public X509Certificate[] getAcceptedIssuers() {  
 return null;  
 }  
  
 @Override  
 public void checkClientTrusted(X509Certificate[] x509Certificates, String s, Socket socket) throws CertificateException {  
  
 }  
  
 @Override  
 public void checkServerTrusted(X509Certificate[] x509Certificates, String s, Socket socket) throws CertificateException {  
  
 }  
  
 @Override  
 public void checkClientTrusted(X509Certificate[] x509Certificates, String s, SSLEngine sslEngine) throws CertificateException {  
  
 }  
  
 @Override  
 public void checkServerTrusted(X509Certificate[] x509Certificates, String s, SSLEngine sslEngine) throws CertificateException {  
  
 }  
 }  
  
 public static void main(String[] args) throws Exception {  
 Long timestamp = new Date().getTime();  
 String appId = "DLc8Qk3bXR2xuI6K";  
 String appSecret = "DaxOC9DIVU7cfAb40sdueb3axYE8Iz";  
  
 String key = appId + "," + timestamp + "," + appSecret;  
  
 String s = SHAEncrypt.SHA512(key);  
 Map<String, String> form = new HashMap<>();  
 form.put("appId", appId);  
 form.put("timestamp", timestamp.toString());  
 form.put("signature", s);  
 form.put("idfa", "89919D5E-8074-4C8E-B629-EFB8A4B3E664");  
 form.put("sign", "8E466DB4517A0A14F4671DD4A355D3B1E550751F7B604599EC490A999A86B31C873F8D4CAEDE9C1B2D34CF7D8CE91017EACD7D57A87AE4FD67D0BEED70990B7E8931C4C2BDC412F01AA6E9C484D11FF51D415BD68B561F496244DF303CEB94FB");  
  
  
 System.***out***.println(timestamp);  
 System.***out***.println(s);  
  
  
 String query = "http://101.201.103.151:8081/sdk/list/get";  
 Long start = System.currentTimeMillis();  
 System.***out***.println(handlePost(query,form));  
 Long end = System.currentTimeMillis();  
 System.***out***.println(end - start);  
}  
  
}

## 2、发送header参数和post参数 请求

### 1、请求文档说明

**接口地址**：<http://shike.console.duodian.com/manage/public/transferBalance>   
**请求方式**：POST   
**请求参数**

| **请求字段** | **字段类型** | **字段描述** |
| --- | --- | --- |
| customerId | Long | 客户端ID(非空) |
| money | BigDecimal | 金额(非空) |

**接口响应**

| **字段** | **字段类型** | **字段描述** |
| --- | --- | --- |
| code | String | 状态：0正常 |
| message | String | 响应描述 |
| result | Array | 客户信息数组 |

客户信息

| **字段** | **字段类型** | **字段描述** |
| --- | --- | --- |
| customerId | Long | 客户ID |
| customerName | String | 公司名称 |
| balance | BigDecimal | 余额 |

code 说明：

| **code值** | **code含义** |
| --- | --- |
| 0 | 正常 |
| 400 | 参数非法 |
| 02002 | 客户余额不足 |

python请求示例：

1. import urllib2
2. import urllib
3. import time
4. import uuid
5. import hashlib
6. URL = 'http://shike.console.duodian.com/manage/public/transferBalance'
7. APPID = 'c99e2a26aac04438b167868d1466388a'
8. APPSCRET = '61dc3866f1a243acbd9e088668c8201a'
9. if \_\_name\_\_ == '\_\_main\_\_':
10. headers = {}
11. headers['timestamp'] = str(int(time.time() \* 1000))
12. headers['requestId'] = str(uuid.uuid4())
13. headers['signature'] = hashlib.sha512(APPID + headers['requestId'] + headers['timestamp'] + APPSCRET).hexdigest()
14. headers['appid']= APPID
15. params = {}
16. params['customerId'] = 3321
17. params['money'] = 7
18. data = urllib.urlencode(params)
19. request = urllib2.Request(URL, data=data, headers=headers)
20. response = urllib2.urlopen(request)
21. print response.read()

响应json示例：

1. {
2. "code": "0",
3. "message": "OK",
4. "result": null
5. }

### 2、header请求中存放的参数有

/\*\*  
 \* 改变老后台余额转移审核状态  
 \* @param id  
 \* @param auditStatus  
 \*/  
@Override  
public void changeStatus(Long id,Integer auditStatus) {  
 if(auditStatus.compareTo(EnumOldBackGroundBalanceTranferStatus.***已通过***.status)==0){  
 OldBackGroundBalanceTransferUser oldBackGroundBalanceTransferUser = oldBackGroundBalanceTransferUserDAO.findOne(id);  
 try {  
 operate(oldBackGroundBalanceTransferUser.getUserId(),oldBackGroundBalanceTransferUser.getBalance(),BigDecimal.valueOf(0));  
   
 getCallBack(oldBackGroundBalanceTransferUser.getCustomerId(),oldBackGroundBalanceTransferUser.getBalance());  
 }catch (Exception e){  
 throw new AppException("新后台增加金额失败");  
 }  
 }  
 oldBackGroundBalanceTransferUserDAO.changeAuditStatus(auditStatus,id);  
}

public void getCallBack(Long customerId,BigDecimal money) {  
  
 SysAdminUser adminUser = AppSessionHelper.getSessionUser();  
  
 Long timestamp = System.currentTimeMillis();  
  
 String appid = "c99e2a26aac04438b167868d1466388a";  
 String appSecret = "61dc3866f1a243acbd9e088668c8201a";  
 String requestId = UUID.randomUUID().toString();  
 String signature = getCallBackSigntureInName(appid,requestId,timestamp.toString(),appSecret);  
  
 Map<String, String> headerParams = new HashMap<>();  
 headerParams.put("timestamp",timestamp.toString());  
 headerParams.put("requestId",requestId);  
 headerParams.put("signature",signature);  
 headerParams.put("appid",appid);  
  
 Map<String, String> formParams = new HashMap<>();  
 formParams.put("customerId",customerId.toString());  
 formParams.put("money",money.toString());  
 formParams.put("operatorId",adminUser.getId().toString());  
 formParams.put("operatorName",adminUser.getName());  
  
  
  
 String url = "http://shike.console.duodian.com/manage/public/transferBalance";  
 callBackhandlePost(url,headerParams,formParams);  
}  
  
  
  
private String getCallBackSigntureInName(String appid, String requestId, String timestamp, String appSecret){  
 logger.info("appid:"+appid+", requestId:"+requestId+", timestamp:"+timestamp+", secret:"+appSecret);  
 return SHAEncrypt.SHA512(appid + requestId + timestamp+ appSecret);  
}  
  
  
  
public void callBackhandlePost(String url, Map<String, String> headerParams,Map<String, String> formParams) {  
 CloseableHttpClient httpclient = HttpClients.custom().build();  
  
 try {  
 HttpPost request = new HttpPost(url);  
 List<NameValuePair> formValue = new ArrayList();  
 Iterator formParamsIterator = formParams.entrySet().iterator();  
 Iterator headderParams = headerParams.entrySet().iterator();  
  
 while (formParamsIterator.hasNext()) {  
 Map.Entry<String, String> entry = (Map.Entry) formParamsIterator.next();  
 formValue.add(new BasicNameValuePair((String) entry.getKey(), (String) entry.getValue()));  
 }  
 while (headderParams.hasNext()){  
 Map.Entry<String, String> entry = (Map.Entry) headderParams.next();  
 request.setHeader(entry.getKey(), entry.getValue());  
 }  
  
 request.setEntity(new UrlEncodedFormEntity(formValue, "UTF-8"));  
 request.setHeader("User-Agent", "admore");  
 request.setConfig(defaultRequestConfig());  
 HttpResponse response = httpclient.execute(request);  
 String result = IOUtils.toString(response.getEntity().getContent(), "UTF-8");  
  
 logger.info("result:"+result);  
 if (StringUtils.isNotBlank(result)){  
 JSONObject json = JSONObject.fromObject(result);  
 String code = json.get("code").toString();  
 if(!"0".equals(code)){  
 logger.info("减去老后台余额失败!"+json.get("message").toString());  
 throw new AppException("减去老后台余额失败");  
 }  
 }  
 } catch (IOException var10) {  
 throw new RuntimeException(var10);  
 } finally {  
 IOUtils.closeQuietly(httpclient);  
 }  
}

# 3、http请求携带证书内容（微信支付）

// 证书 密码默认是商户号  
char[] password = keyStorePassword.toCharArray();  
  
ClassPathResource resource = new ClassPathResource("cert/wechat/apiclient\_cert.p12");  
InputStream certinputStream = resource.getInputStream();  
KeyStore ks = KeyStore.getInstance("PKCS12");  
ks.load(certinputStream, password);  
  
// 实例化密钥库 & 初始化密钥工厂  
KeyManagerFactory kmf = KeyManagerFactory.getInstance(KeyManagerFactory.getDefaultAlgorithm());  
kmf.init(ks, password);  
  
// 创建 SSLContext  
SSLContext sslContext = SSLContext.getInstance("TLS");  
sslContext.init(kmf.getKeyManagers(), null, new SecureRandom());  
SSLSocketFactory ssf = sslContext.getSocketFactory();

/\*\*  
 \* 发送https请求  
 \*  
 \* @param requestUrl 请求地址  
 \* @param method 请求方式（GET、POST）  
 \* @param xmlParam 提交的数据  
 \* @return 返回微信服务器响应的信息  
 \*/  
public static String httpsRequest(String requestUrl, String method, String xmlParam,String keyStorePassword) {  
 try  
 {  
 // 证书 密码默认是商户号  
 char[] password = keyStorePassword.toCharArray();  
  
 ClassPathResource resource = new ClassPathResource("cert/wechat/apiclient\_cert.p12");  
 InputStream certinputStream = resource.getInputStream();  
 KeyStore ks = KeyStore.getInstance("PKCS12");  
 ks.load(certinputStream, password);  
  
 // 实例化密钥库 & 初始化密钥工厂  
 KeyManagerFactory kmf = KeyManagerFactory.getInstance(KeyManagerFactory.getDefaultAlgorithm());  
 kmf.init(ks, password);  
  
 // 创建 SSLContext  
 SSLContext sslContext = SSLContext.getInstance("TLS");  
 sslContext.init(kmf.getKeyManagers(), null, new SecureRandom());  
 SSLSocketFactory ssf = sslContext.getSocketFactory();  
  
  
  
 URL url = new URL(requestUrl);  
 HttpsURLConnection conn = (HttpsURLConnection) url.openConnection();  
 conn.setSSLSocketFactory(ssf); //证书  
 conn.setDoOutput(true);  
 conn.setDoInput(true);  
 conn.setUseCaches(false);  
 // 设置请求方式（GET/POST）  
 conn.setRequestMethod(method);  
 conn.setRequestProperty("content-type", "application/x-www-form-urlencoded");  
 // 当outputStr不为null时向输出流写数据  
 if (null != xmlParam)  
 {  
 OutputStream outputStream = conn.getOutputStream();  
 // 注意编码格式  
 outputStream.write(xmlParam.getBytes("UTF-8"));  
 outputStream.close();  
 }  
 // 从输入流读取返回内容  
 InputStream inputStream = conn.getInputStream();  
 InputStreamReader inputStreamReader = new InputStreamReader(inputStream, "UTF-8");  
 BufferedReader bufferedReader = new BufferedReader(inputStreamReader);  
 String str = null;  
 StringBuffer buffer = new StringBuffer();  
 while ((str = bufferedReader.readLine()) != null) {  
 buffer.append(str);  
 }  
 // 释放资源  
 bufferedReader.close();  
 inputStreamReader.close();  
 inputStream.close();  
 inputStream = null;  
 conn.disconnect();  
 return buffer.toString();  
 } catch (ConnectException ce) {  
 ExceptionLogUtils.log(ce,PayCommonHttpXmlUtil.class );  
 } catch (IOException e) {  
 ExceptionLogUtils.log(e,PayCommonHttpXmlUtil.class );  
 }catch (Exception e) {  
 ExceptionLogUtils.log(e,PayCommonHttpXmlUtil.class );  
 }  
 return null;  
}