# http Get方式请求：

**public** **static** String sendGetHttps(String url, Map<String, String> params,

String... endPath) **throws** Exception {

url = *appendParamsToUrl*(url, params, endPath);

System.***out***.println(url);

URL requestUrl = **new** URL(url);

HttpsURLConnection con = HttpsURLConnection.**class**.cast(requestUrl

.openConnection());

con.setRequestMethod("GET");

con.connect();

InputStream input = con.getInputStream();

String result = "";

**byte**[] buffer = **new** **byte**[1024 \* 8];

**int** index = -1;

**while** ((index = input.read(buffer)) != -1) {

result += **new** String(buffer, 0, index,"UTF-8");

}

input.close();

con.disconnect();

**return** result;

}

|  |
| --- |
| /\*\*  \* 将参数拼接到url后边  \*  \* **@param** url  \* **@param** params  \* **@param** endPath  \* 末端自定义path  \* **@return**  \*/  **public** **static** String appendParamsToUrl(String url,  Map<String, String> params, String... endPath) {  url += "?";  **int** i = 0;  **for** (String key : params.keySet()) {  url += i == 0 ? key + "=" + params.get(key) : "&" + key + "="  + params.get(key);  i++;  }  **if** (endPath != **null** && endPath.length != 0) {  **for** (**int** j = 0; j < endPath.length; j++) {  url += endPath[j];  }  }  **return** url;  } |

# 发送https get请求

/\*\*

\* 发送https get请求

\*

\* **@param** url

\* 请求URL

\* **@param** params

\* 请求参数

\* **@param** header

\* 请求头

\* **@param** endPath

\* URL末端拼接串

\* **@return**

\* **@throws** Exception

\*/

**public** **static** String sendGetHttps(String url, Map<String, String> params,

Map<String, String> header, String... endPath) **throws** Exception {

// 忽略证书

//trustAllHttpsCertificates();

//HttpsURLConnection.setDefaultHostnameVerifier(hv);

url = *appendParamsToUrl*(url, params, endPath);

System.***out***.println(url);

URL requestUrl = **new** URL(url);

HttpsURLConnection con = HttpsURLConnection.**class**.cast(requestUrl

.openConnection());

con.setRequestMethod("GET");

**for** (Entry<String, String> entry : header.entrySet()) {

con.addRequestProperty(entry.getKey(), entry.getValue());

}

con.connect();

InputStream input = con.getInputStream();

String result = "";

**byte**[] buffer = **new** **byte**[1024 \* 8];

**int** index = -1;

**while** ((index = input.read(buffer)) != -1) {

result += **new** String(buffer, 0, index);

}

input.close();

con.disconnect();

**return** result;

}

# 发送http post请求：

/\*\*

\* 发送Post 请求

\*

\* **@param** url

\* **@param** params

\* **@param** header

\* **@param** endPath

\* **@return** 返回状态码

\* **@throws** Exception

\*/

**public** **static** **int** sendPostRequest(String url, Map<String, String> params,

Map<String, String> header, String... endPath) **throws** Exception {

url = *appendParamsToUrl*(url, params, endPath);

System.***out***.println(url);

URL requestUrl = **new** URL(url);

HttpsURLConnection con = HttpsURLConnection.**class**.cast(requestUrl

.openConnection());

con.setRequestMethod("POST");

**for** (Entry<String, String> item : header.entrySet()) {

con.setRequestProperty(item.getKey(), item.getValue());

}

con.connect();

**int** code = con.getResponseCode();

con.disconnect();

**return** code;

}

# 发送https post请求

/\*\*

\* 发送Post 请求

\*

\* **@param** url

\* 地址

\* **@param** params

\* 参数

\* **@param** header

\* 头

\* **@param** data

\* 数据

\* **@param** endPath

\* **@throws** Exception

\*/

**public** **static** String sendPostHttps(String url, Map<String, String> params,

Map<String, String> header, String data, String... endPath)

**throws** Exception {

// 忽略证书

//trustAllHttpsCertificates();

//HttpsURLConnection.setDefaultHostnameVerifier(hv);

url = *appendParamsToUrl*(url, params, endPath);

System.***err***.println(url);

URL requestUrl = **new** URL(url);

HttpsURLConnection con = HttpsURLConnection.**class**.cast(requestUrl

.openConnection());

con.setDoInput(**true**);

con.setUseCaches(**false**);

con.setDoOutput(**true**);

con.setRequestMethod("POST");

**for** (Entry<String, String> item : header.entrySet()) {

con.setRequestProperty(item.getKey(), item.getValue());

}

con.connect();

**if** (data != **null**) {

**byte**[] dataBytes = data.getBytes("UTF-8");

OutputStream out = con.getOutputStream();

out.write(dataBytes);

out.flush();

out.close();

}

InputStream input = con.getInputStream();

StringBuilder sb = **new** StringBuilder();

**int** i = -1;

**byte**[] buffer = **new** **byte**[1024 \* 8];

**while** ((i = input.read(buffer)) != -1) {

sb.append(**new** String(buffer, 0, i));

}

input.close();

**return** sb.toString();

}

# 发送https post请求不带请求头

/\*\*

\* 发送https post 请求不带请求头

\*

\* **@param** url

\* **@param** params

\* **@param** data

\* **@param** endPath

\* **@return**

\* **@throws** Exception

\*/

**public** **static** String sendPostHttps(String url, Map<String, String> params,

String data, String... endPath) **throws** Exception {

url = *appendParamsToUrl*(url, params, endPath);

URL requestUrl = **new** URL(url);

HttpsURLConnection con = HttpsURLConnection.**class**.cast(requestUrl

.openConnection());

con.setRequestMethod("POST");

con.setDoInput(**true**);

con.setDoOutput(**true**);

con.setUseCaches(**false**);

con.connect();

**if** (data != **null**) {

**byte**[] dataBytes = data.getBytes("UTF-8");

OutputStream out = con.getOutputStream();

out.write(dataBytes);

out.flush();

out.close();

}

InputStream input = con.getInputStream();

StringBuilder sb = **new** StringBuilder();

**int** i = -1;

**byte**[] buffer = **new** **byte**[1024 \* 8];

**while** ((i = input.read(buffer)) != -1) {

sb.append(**new** String(buffer, 0, i));

}

input.close();

**return** sb.toString();

}

# 将参数拼接到url后边

/\*\*

\* 将参数拼接到url后边

\*

\* **@param** url

\* **@param** params

\* **@param** endPath

\* 末端自定义path

\* **@return**

\*/

**public** **static** String appendParamsToUrl(String url,

Map<String, String> params, String... endPath) {

**if**(params==**null**||params.size()==0){

**return** url;

}

url += "?";

**int** i = 0;

**for** (String key : params.keySet()) {

url += i == 0 ? key + "=" + params.get(key) : "&" + key + "="

+ params.get(key);

i++;

}

**if** (endPath != **null** && endPath.length != 0) {

**for** (**int** j = 0; j < endPath.length; j++) {

url += endPath[j];

}

}

**return** url;

}

}

# Xml转换工具类（此类需要jdomjar包的支持）



## 将对象转成element对象

**public** **static** Element ObjectToXml(Object t){

Class cs=t.getClass();

Field[] fs=cs.getDeclaredFields();

Element root=**new** Element(cs.getSimpleName());

**for**(Field f: fs){

**if**(f.getAnnotation(NotRequired.**class**)==**null**){

**try** {

String methodName="get"+f.getName().substring(0,1).toUpperCase()+f.getName().substring(1);

Method method=cs.getMethod(methodName);

root.addContent(**new** Element(f.getName().toLowerCase()).setContent(**new** CDATA(String.*valueOf*(method.invoke(t)))));

}**catch** (Exception e) {

*log*.error(e);

}

}

}

**return** root;

}

## 将map转成element对象

**public** **static** Element mapToXml(Map<String,String> params){

Element root=**new** Element("xml");

**for** (Entry<String, String> item : params.entrySet()) {

root.addContent(**new** Element(item.getKey()).setContent(**new** CDATA(item.getValue())));

}

**return** root;

}

## 将element对象转成字符串

**public** **static** String xmlToString(Element ele){

ByteArrayOutputStream out=**new** ByteArrayOutputStream();

XMLOutputter xmlOut=**new** XMLOutputter(*format*);

**try** {

xmlOut.output(ele, out);

**return** out.toString();

} **catch** (IOException e) {

e.printStackTrace();

**return** **null**;

}

}

# HttpClient发送带请求头的url（带签名的）

|  |
| --- |
| **package** cn.easier.woact.utils;  **import** java.io.BufferedReader;  **import** java.io.InputStreamReader;  **import** java.io.UnsupportedEncodingException;  **import** java.security.MessageDigest;  **import** java.text.SimpleDateFormat;  **import** java.util.Date;  **import** net.sf.json.JSONObject;  **import** org.apache.http.HttpEntity;  **import** org.apache.http.HttpResponse;  **import** org.apache.http.client.HttpClient;  **import** org.apache.http.client.methods.HttpPost;  **import** org.apache.http.entity.StringEntity;  **import** org.apache.http.impl.client.DefaultHttpClient;  @SuppressWarnings("deprecation")  **public** **class** HttpClientUtil {  **private** **static** String clientId = WebUtils.getConfigProperties("clientId");    **private** **static** String clientSecret = WebUtils.getConfigProperties("clientSecret");    **public** **static** String sendPost(String url,JSONObject data) **throws** Exception{  String timestamp = **new** SimpleDateFormat("yyyy-MM-dd HH:mm:ss").format(**new** Date());  HttpClient client = getResource();  HttpPost post = **new** HttpPost(url);  post.setHeader("Content-Encoding", "UTF-8");  post.setHeader("Content-Type", "text/plain");  post.setHeader("Date", timestamp);  post.setHeader("Authorization", clientId + ":" + postSignStr(timestamp));    StringEntity s = **null**;  **try** {  s = **new** StringEntity(data.toString());  } **catch** (UnsupportedEncodingException e) {  e.printStackTrace();  }  s.setContentEncoding("UTF-8");  s.setContentType("text/plain");  post.setEntity(s);    HttpResponse response = client.execute(post);  HttpEntity entity = response.getEntity();  BufferedReader reader = **new** BufferedReader(**new** InputStreamReader(entity.getContent(), "UTF-8"));  String result="";  String line = **null**;  **while** ((line = reader.readLine()) != **null**) {  result=result+line;  }    **return** result;  }    **private** **static** HttpClient getResource(){  DefaultHttpClient client = **new** DefaultHttpClient();  **return** client;  }    **private** **static** String postSignStr(String timestamp){  String signStr = "Date=" + timestamp;  signStr = signStr + clientSecret;  **try** {  signStr = md5Encode(signStr).toUpperCase();  } **catch** (Exception e) {  e.printStackTrace();  }  **return** signStr;  }  **private** **static** String md5Encode(String inStr) **throws** Exception {  MessageDigest md5 = **null**;  **try** {  md5 = MessageDigest.getInstance("MD5");  } **catch** (Exception e) {  System.out.println(e.toString());  e.printStackTrace();  **return** "";  }  **byte**[] byteArray = inStr.getBytes("UTF-8");  **byte**[] md5Bytes = md5.digest(byteArray);  StringBuffer hexValue = **new** StringBuffer();  **for** (**int** i = 0; i < md5Bytes.length; i++) {  **int** val = ((**int**) md5Bytes[i]) & 0xff;  **if** (val < 16) {  hexValue.append("0");  }  hexValue.append(Integer.toHexString(val));  }  **return** hexValue.toString();  }  } |