httpClient 工具类

作者：MiyaJ
原文链接：https://ld246.com/article/1540898633453
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)

```
package com.utils;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.net.HttpURLConnection;
import java.net.InetSocketAddress;
import java.net.Proxy;
import java.net.URL;

import org.apache.http.HttpEntity;
import org.apache.http.HttpStatus;
import org.apache.http.client.config.RequestConfig;
import org.apache.http.client.methods.CloseableHttpResponse;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClients;

public class HttpUtils {
    private static String proxyHost = "127.0.0.1";
    private static int proxyPort = 8087;


    public final static String get(String url, String params) {
        String response = null;
        // 拼接请求URL
        // 拼接后的URL形如"http://****.cn/getSpecifyCategoryNews?start=1&count=5&id=23"
        if (null != params && !params.equals(""))
            url += "?" + params;

        // 创造HttpClient的实例
        CloseableHttpClient httpClient = HttpClients.createDefault();
        // 创建GET方法的实例
        HttpGet httpGet = new HttpGet(url);
        // 设置请求和传输超时时间
        int timeoutConnection = 10000;
        int timeoutSocket = 20000;
        RequestConfig requestConfig = RequestConfig.custom().setSocketTimeout(timeoutSock
t).setConnectTimeout(timeoutConnection).build();
        httpGet.setConfig(requestConfig);
        try {
            CloseableHttpResponse httpResponse = httpClient.execute(httpGet);
            int statusCode = httpResponse.getStatusLine().getStatusCode();
            if (statusCode == HttpStatus.SC_OK) // SC_OK = 200
            {
                try {
                    // 获得服务器响应的消息体（不包括http head）
                    HttpEntity entity = httpResponse.getEntity();
                    if (entity != null) {
                        InputStream is = entity.getContent();
                        // 获得响应字符集编码
                        // ContentType contentType =
```

```
                // ContentType.getOrDefault(entity);
                // Charset charset = contentType.getCharset();
                // 将inputstream转化为reader，并使用缓冲按行读取内容
                // BufferedReader br = new BufferedReader(new
                // InputStreamReader(is,charset));
                BufferedReader br = new BufferedReader(new InputStreamReader(is, "UTF-8"))

                StringBuffer resBuffer = new StringBuffer();
                String line = null;
                while ((line = br.readLine()) != null)
                    resBuffer.append(line + "\r\n");
                response = resBuffer.toString();
                is.close();// 关闭响应实体内容流
            }
        } finally {// 关闭响应
            httpResponse.close();
        }
    } else
        response = "返回码：" + statusCode;
  } catch (Exception e) {
    e.printStackTrace();
  } finally {
    try {// 关闭连接
        httpClient.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
  }
  return response;
}

/**
 * 向指定 URL 发送POST方法的请求
 *
 * @param url
 *          发送请求的 URL
 * @param param
 *          请求参数，请求参数应该是 name1=value1&name2=value2 的形式。
 * @param isproxy
 *          是否使用代理模式
 * @return 所代表远程资源的响应结果
 */
public static String sendPost(String url, String param) {
  OutputStreamWriter out = null;
  BufferedReader in = null;
  String result = "";
  try {
    URL realUrl = new URL(url);
    HttpURLConnection conn = null;
    // 打开和URL之间的连接
    conn = (HttpURLConnection) realUrl.openConnection();

    // 发送POST请求必须设置如下两行
    conn.setDoOutput(true);
```

```
            conn.setDoInput(true);
            conn.setRequestMethod("POST"); // POST方法

            // 设置通用的请求属性

            conn.setRequestProperty("accept", "*/*");
            conn.setRequestProperty("connection", "Keep-Alive");
            conn.setRequestProperty("user-agent", "Mozilla/4.0 (compatible; MSIE 6.0; Windows
T 5.1;SV1)");
            conn.setRequestProperty("Content-Type", "application/x-www-form-urlencoded");

            conn.connect();

            // 获取URLConnection对象对应的输出流
            out = new OutputStreamWriter(conn.getOutputStream(), "UTF-8");
            // 发送请求参数
            out.write(param);
            // flush输出流的缓冲
            out.flush();
            // 定义BufferedReader输入流来读取URL的响应
            in = new BufferedReader(new InputStreamReader(conn.getInputStream()));
            String line;
            while ((line = in.readLine()) != null) {
                result += line;
            }
        } catch (Exception e) {
            System.out.println("发送 POST 请求出现异常！" + e);
            e.printStackTrace();
        }
        // 使用finally块来关闭输出流、输入流
        finally {
            try {
                if (out != null) {
                    out.close();
                }
                if (in != null) {
                    in.close();
                }
            } catch (IOException ex) {
                ex.printStackTrace();
            }
        }
        return result;
    }

    /**
     * 向指定 URL 发送POST方法的请求
     *
     * @param url
     *            发送请求的 URL
     * @param param
     *            请求参数，请求参数应该是 name1=value1&name2=value2 的形式。
     * @param isproxy
     *            是否使用代理模式
```

```
     * @return 所代表远程资源的响应结果
     */
    public static String sendPost(String url, String param, boolean isproxy) {
        OutputStreamWriter out = null;
        BufferedReader in = null;
        String result = "";
        try {
            URL realUrl = new URL(url);
            HttpURLConnection conn = null;
            if (isproxy) {// 使用代理模式
                @SuppressWarnings("static-access")
                Proxy proxy = new Proxy(Proxy.Type.DIRECT.HTTP, new InetSocketAddress(proxyHo
t, proxyPort));
                conn = (HttpURLConnection) realUrl.openConnection(proxy);
            } else {
                conn = (HttpURLConnection) realUrl.openConnection();
            }
            // 打开和URL之间的连接

            // 发送POST请求必须设置如下两行
            conn.setDoOutput(true);
            conn.setDoInput(true);
            conn.setRequestMethod("POST"); // POST方法

            // 设置通用的请求属性

            conn.setRequestProperty("accept", "*/*");
            conn.setRequestProperty("connection", "Keep-Alive");
            conn.setRequestProperty("user-agent", "Mozilla/4.0 (compatible; MSIE 6.0; Windows
T 5.1;SV1)");
            conn.setRequestProperty("Content-Type", "application/x-www-form-urlencoded");

            conn.connect();

            // 获取URLConnection对象对应的输出流
            out = new OutputStreamWriter(conn.getOutputStream(), "UTF-8");
            // 发送请求参数
            out.write(param);
            // flush输出流的缓冲
            out.flush();
            // 定义BufferedReader输入流来读取URL的响应
            in = new BufferedReader(new InputStreamReader(conn.getInputStream()));
            String line;
            while ((line = in.readLine()) != null) {
                result += line;
            }
        } catch (Exception e) {
            System.out.println("发送 POST 请求出现异常！" + e);
            e.printStackTrace();
        }
        // 使用finally块来关闭输出流、输入流
        finally {
            try {
                if (out != null) {
```

```
                out.close();
            }
            if (in != null) {
                in.close();
            }
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }
    return result;
  }
}
```