[**HttpClient4.5 简单入门实例(一)**](http://tzz6.iteye.com/blog/2224757)

**博客分类：**

* [httpclient](http://tzz6.iteye.com/category/342464)

[httpclient](http://www.iteye.com/blogs/tag/httpclient)[get](http://www.iteye.com/blogs/tag/get)[post](http://www.iteye.com/blogs/tag/post)[http](http://www.iteye.com/blogs/tag/http)

一、所需要的jar包

   httpclient-4.5.jar

   httpcore-4.4.1.jar

   httpmime-4.5.jar

二、实例

**Java代码  [收藏代码](javascript:void())**

1. **package** cn.tzz.apache.httpclient;

4. **import** java.io.File;
5. **import** java.io.IOException;
6. **import** java.net.URL;
7. **import** java.util.ArrayList;
8. **import** java.util.List;
9. **import** java.util.Map;
11. **import** org.apache.http.HttpEntity;
12. **import** org.apache.http.NameValuePair;
13. **import** org.apache.http.client.config.RequestConfig;
14. **import** org.apache.http.client.entity.UrlEncodedFormEntity;
15. **import** org.apache.http.client.methods.CloseableHttpResponse;
16. **import** org.apache.http.client.methods.HttpGet;
17. **import** org.apache.http.client.methods.HttpPost;
18. **import** org.apache.http.conn.ssl.DefaultHostnameVerifier;
19. **import** org.apache.http.conn.util.PublicSuffixMatcher;
20. **import** org.apache.http.conn.util.PublicSuffixMatcherLoader;
21. **import** org.apache.http.entity.ContentType;
22. **import** org.apache.http.entity.StringEntity;
23. **import** org.apache.http.entity.mime.MultipartEntityBuilder;
24. **import** org.apache.http.entity.mime.content.FileBody;
25. **import** org.apache.http.entity.mime.content.StringBody;
26. **import** org.apache.http.impl.client.CloseableHttpClient;
27. **import** org.apache.http.impl.client.HttpClients;
28. **import** org.apache.http.message.BasicNameValuePair;
29. **import** org.apache.http.util.EntityUtils;

32. **public** **class** HttpClientUtil {
33. **private** RequestConfig requestConfig = RequestConfig.custom()
34. .setSocketTimeout(15000)
35. .setConnectTimeout(15000)
36. .setConnectionRequestTimeout(15000)
37. .build();
39. **private** **static** HttpClientUtil instance = **null**;
40. **private** HttpClientUtil(){}
41. **public** **static** HttpClientUtil getInstance(){
42. **if** (instance == **null**) {
43. instance = **new** HttpClientUtil();
44. }
45. **return** instance;
46. }
48. /\*\*
49. \* 发送 post请求
50. \* @param httpUrl 地址
51. \*/
52. **public** String sendHttpPost(String httpUrl) {
53. HttpPost httpPost = **new** HttpPost(httpUrl);// 创建httpPost
54. **return** sendHttpPost(httpPost);
55. }
57. /\*\*
58. \* 发送 post请求
59. \* @param httpUrl 地址
60. \* @param params 参数(格式:key1=value1&key2=value2)
61. \*/
62. **public** String sendHttpPost(String httpUrl, String params) {
63. HttpPost httpPost = **new** HttpPost(httpUrl);// 创建httpPost
64. **try** {
65. //设置参数
66. StringEntity stringEntity = **new** StringEntity(params, "UTF-8");
67. stringEntity.setContentType("application/x-www-form-urlencoded");
68. httpPost.setEntity(stringEntity);
69. } **catch** (Exception e) {
70. e.printStackTrace();
71. }
72. **return** sendHttpPost(httpPost);
73. }
75. /\*\*
76. \* 发送 post请求
77. \* @param httpUrl 地址
78. \* @param maps 参数
79. \*/
80. **public** String sendHttpPost(String httpUrl, Map<String, String> maps) {
81. HttpPost httpPost = **new** HttpPost(httpUrl);// 创建httpPost
82. // 创建参数队列
83. List<NameValuePair> nameValuePairs = **new** ArrayList<NameValuePair>();
84. **for** (String key : maps.keySet()) {
85. nameValuePairs.add(**new** BasicNameValuePair(key, maps.get(key)));
86. }
87. **try** {
88. httpPost.setEntity(**new** UrlEncodedFormEntity(nameValuePairs, "UTF-8"));
89. } **catch** (Exception e) {
90. e.printStackTrace();
91. }
92. **return** sendHttpPost(httpPost);
93. }

96. /\*\*
97. \* 发送 post请求（带文件）
98. \* @param httpUrl 地址
99. \* @param maps 参数
100. \* @param fileLists 附件
101. \*/
102. **public** String sendHttpPost(String httpUrl, Map<String, String> maps, List<File> fileLists) {
103. HttpPost httpPost = **new** HttpPost(httpUrl);// 创建httpPost
104. MultipartEntityBuilder meBuilder = MultipartEntityBuilder.create();
105. **for** (String key : maps.keySet()) {
106. meBuilder.addPart(key, **new** StringBody(maps.get(key), ContentType.TEXT\_PLAIN));
107. }
108. **for**(File file : fileLists) {
109. FileBody fileBody = **new** FileBody(file);
110. meBuilder.addPart("files", fileBody);
111. }
112. HttpEntity reqEntity = meBuilder.build();
113. httpPost.setEntity(reqEntity);
114. **return** sendHttpPost(httpPost);
115. }
117. /\*\*
118. \* 发送Post请求
119. \* @param httpPost
120. \* @return
121. \*/
122. **private** String sendHttpPost(HttpPost httpPost) {
123. CloseableHttpClient httpClient = **null**;
124. CloseableHttpResponse response = **null**;
125. HttpEntity entity = **null**;
126. String responseContent = **null**;
127. **try** {
128. // 创建默认的httpClient实例.
129. httpClient = HttpClients.createDefault();
130. httpPost.setConfig(requestConfig);
131. // 执行请求
132. response = httpClient.execute(httpPost);
133. entity = response.getEntity();
134. responseContent = EntityUtils.toString(entity, "UTF-8");
135. } **catch** (Exception e) {
136. e.printStackTrace();
137. } **finally** {
138. **try** {
139. // 关闭连接,释放资源
140. **if** (response != **null**) {
141. response.close();
142. }
143. **if** (httpClient != **null**) {
144. httpClient.close();
145. }
146. } **catch** (IOException e) {
147. e.printStackTrace();
148. }
149. }
150. **return** responseContent;
151. }
153. /\*\*
154. \* 发送 get请求
155. \* @param httpUrl
156. \*/
157. **public** String sendHttpGet(String httpUrl) {
158. HttpGet httpGet = **new** HttpGet(httpUrl);// 创建get请求
159. **return** sendHttpGet(httpGet);
160. }
162. /\*\*
163. \* 发送 get请求Https
164. \* @param httpUrl
165. \*/
166. **public** String sendHttpsGet(String httpUrl) {
167. HttpGet httpGet = **new** HttpGet(httpUrl);// 创建get请求
168. **return** sendHttpsGet(httpGet);
169. }
171. /\*\*
172. \* 发送Get请求
173. \* @param httpPost
174. \* @return
175. \*/
176. **private** String sendHttpGet(HttpGet httpGet) {
177. CloseableHttpClient httpClient = **null**;
178. CloseableHttpResponse response = **null**;
179. HttpEntity entity = **null**;
180. String responseContent = **null**;
181. **try** {
182. // 创建默认的httpClient实例.
183. httpClient = HttpClients.createDefault();
184. httpGet.setConfig(requestConfig);
185. // 执行请求
186. response = httpClient.execute(httpGet);
187. entity = response.getEntity();
188. responseContent = EntityUtils.toString(entity, "UTF-8");
189. } **catch** (Exception e) {
190. e.printStackTrace();
191. } **finally** {
192. **try** {
193. // 关闭连接,释放资源
194. **if** (response != **null**) {
195. response.close();
196. }
197. **if** (httpClient != **null**) {
198. httpClient.close();
199. }
200. } **catch** (IOException e) {
201. e.printStackTrace();
202. }
203. }
204. **return** responseContent;
205. }
207. /\*\*
208. \* 发送Get请求Https
209. \* @param httpPost
210. \* @return
211. \*/
212. **private** String sendHttpsGet(HttpGet httpGet) {
213. CloseableHttpClient httpClient = **null**;
214. CloseableHttpResponse response = **null**;
215. HttpEntity entity = **null**;
216. String responseContent = **null**;
217. **try** {
218. // 创建默认的httpClient实例.
219. PublicSuffixMatcher publicSuffixMatcher = PublicSuffixMatcherLoader.load(**new** URL(httpGet.getURI().toString()));
220. DefaultHostnameVerifier hostnameVerifier = **new** DefaultHostnameVerifier(publicSuffixMatcher);
221. httpClient = HttpClients.custom().setSSLHostnameVerifier(hostnameVerifier).build();
222. httpGet.setConfig(requestConfig);
223. // 执行请求
224. response = httpClient.execute(httpGet);
225. entity = response.getEntity();
226. responseContent = EntityUtils.toString(entity, "UTF-8");
227. } **catch** (Exception e) {
228. e.printStackTrace();
229. } **finally** {
230. **try** {
231. // 关闭连接,释放资源
232. **if** (response != **null**) {
233. response.close();
234. }
235. **if** (httpClient != **null**) {
236. httpClient.close();
237. }
238. } **catch** (IOException e) {
239. e.printStackTrace();
240. }
241. }
242. **return** responseContent;
243. }
244. }

 三、测试代码

**Java代码  [收藏代码](javascript:void())**

1. **package** cn.tzz.apache.httpclient;
3. **import** java.io.File;
4. **import** java.util.ArrayList;
5. **import** java.util.HashMap;
6. **import** java.util.List;
7. **import** java.util.Map;
9. **import** org.junit.Test;
11. **public** **class** HttpClientUtilTest {
13. @Test
14. **public** **void** testSendHttpPost1() {
15. String responseContent = HttpClientUtil.getInstance()
16. .sendHttpPost("http://localhost:8089/test/send?username=test01&password=123456");
17. System.out.println("reponse content:" + responseContent);
18. }
20. @Test
21. **public** **void** testSendHttpPost2() {
22. String responseContent = HttpClientUtil.getInstance()
23. .sendHttpPost("http://localhost:8089/test/send", "username=test01&password=123456");
24. System.out.println("reponse content:" + responseContent);
25. }
27. @Test
28. **public** **void** testSendHttpPost3() {
29. Map<String, String> maps = **new** HashMap<String, String>();
30. maps.put("username", "test01");
31. maps.put("password", "123456");
32. String responseContent = HttpClientUtil.getInstance()
33. .sendHttpPost("http://localhost:8089/test/send", maps);
34. System.out.println("reponse content:" + responseContent);
35. }
36. @Test
37. **public** **void** testSendHttpPost4() {
38. Map<String, String> maps = **new** HashMap<String, String>();
39. maps.put("username", "test01");
40. maps.put("password", "123456");
41. List<File> fileLists = **new** ArrayList<File>();
42. fileLists.add(**new** File("D://test//httpclient//1.png"));
43. fileLists.add(**new** File("D://test//httpclient//1.txt"));
44. String responseContent = HttpClientUtil.getInstance()
45. .sendHttpPost("http://localhost:8089/test/sendpost/file", maps, fileLists);
46. System.out.println("reponse content:" + responseContent);
47. }
49. @Test
50. **public** **void** testSendHttpGet() {
51. String responseContent = HttpClientUtil.getInstance()
52. .sendHttpGet("http://localhost:8089/test/send?username=test01&password=123456");
53. System.out.println("reponse content:" + responseContent);
54. }
56. @Test
57. **public** **void** testSendHttpsGet() {
58. String responseContent = HttpClientUtil.getInstance()
59. .sendHttpsGet("https://www.baidu.com");
60. System.out.println("reponse content:" + responseContent);
61. }
63. }

**Java代码  [收藏代码](javascript:void())**

1. @RequestMapping(value = "/test/send")
2. @ResponseBody
3. **public** Map<String, String> sendPost(HttpServletRequest request) {
4. Map<String, String> maps = **new** HashMap<String, String>();
5. String username = request.getParameter("username");
6. String password = request.getParameter("password");
7. maps.put("username", username);
8. maps.put("password", password);
9. **return** maps;
10. }
12. @RequestMapping(value = "/test/sendpost/file",method=RequestMethod.POST)
13. @ResponseBody
14. **public** Map<String, String> sendPostFile(@RequestParam("files") MultipartFile [] files,HttpServletRequest request) {
15. Map<String, String> maps = **new** HashMap<String, String>();
16. String username = request.getParameter("username");
17. String password = request.getParameter("password");
18. maps.put("username", username);
19. maps.put("password", password);
21. **try** {
22. **for**(MultipartFile file : files){
23. String fileName = file.getOriginalFilename();
24. fileName = **new** String(fileName.getBytes(),"UTF-8");
25. InputStream is = file.getInputStream();
26. **if** (fileName != **null** && !("".equals(fileName))) {
27. File directory = **new** File("D://test//httpclient//file");
28. **if** (!directory.exists()) {
29. directory.mkdirs();
30. }
31. String filePath = ("D://test//httpclient//file") + File.separator + fileName;
32. FileOutputStream fos = **new** FileOutputStream(filePath);
33. **byte**[] buffer = **new** **byte**[1024];
34. **while** (is.read(buffer) > 0) {
35. fos.write(buffer, 0, buffer.length);
36. }
37. fos.flush();
38. fos.close();
39. maps.put("file--"+fileName, "uploadSuccess");
40. }
41. }
42. } **catch** (Exception e) {
43. e.printStackTrace();
44. }
45. **return** maps;
46. }