Apache HttpClient使用详解

# 简介

HttpClient是Apache Jakarta Common下的子项目，用来提供高效的、最新的、功能丰富的支持HTTP协议的客户端编程工具包，并且它支持HTTP协议最新的版本和建议。HttpClient已经应用在很多的项目中，比如Apache Jakarta上很著名的另外两个开源项目Cactus和HTMLUnit都使用了HttpClient。

# 特性

|  |
| --- |
| 1. 基于标准、纯净的java语言。实现了Http1.0和Http1.1  2. 以可扩展的面向对象的结构实现了Http全部的方法（GET, POST, PUT, DELETE, HEAD, OPTIONS, and TRACE）。  3. 支持HTTPS协议。  4. 通过Http代理建立透明的连接。  5. 利用CONNECT方法通过Http代理建立隧道的https连接。  6. Basic, Digest, NTLMv1, NTLMv2, NTLM2 Session, SNPNEGO/Kerberos认证方案。  7. 插件式的自定义认证方案。  8. 便携可靠的套接字工厂使它更容易的使用第三方解决方案。  9. 连接管理器支持多线程应用。支持设置最大连接数，同时支持设置每个主机的最大连接数，发现并关闭过期的连接。  10. 自动处理Set-Cookie中的Cookie。  11. 插件式的自定义Cookie策略。  12. Request的输出流可以避免流中内容直接缓冲到socket服务器。  13. Response的输入流可以有效的从socket服务器直接读取相应内容。  14. 在http1.0和http1.1中利用KeepAlive保持持久连接。  15. 直接获取服务器发送的response code和 headers。  16. 设置连接超时的能力。  17. 实验性的支持http1.1 response caching。  18. 源代码基于Apache License 可免费获取 |

# 使用方法

maven仓库地址：

|  |
| --- |
| <dependency>  <groupId>org.apache.httpcomponents</groupId>  <artifactId>httpclient</artifactId>  <version>4.5.2</version>  </dependency> |

使用HttpClient发送请求、接收响应很简单，一般需要如下几步即可。

第一步： 创建HttpClient对象。

第二步：创建请求方法的实例，并指定请求URL。如果需要发送GET请求，创建HttpGet对象；如果需要发送POST请求，创建HttpPost对象。

第三步：如果需要发送请求参数，可调用HttpGet、HttpPost共同的setParams(HetpParams params)方法来添加请求参数；对于HttpPost对象而言，也可调用setEntity(HttpEntity entity)方法来设置请求参数。

第四步：调用HttpClient对象的execute(HttpUriRequest request)发送请求，该方法返回一个HttpResponse。

第五步：调用HttpResponse的getAllHeaders()、getHeaders(String name)等方法可获取服务器的响应头；调用HttpResponse的getEntity()方法可获取HttpEntity对象，该对象包装了服务器的响应内容。程序可通过该对象获取服务器的响应内容。

第六步：释放连接。无论执行方法是否成功，都必须释放连接

# HttpClient常用API

|  |
| --- |
| #1、创建HTTP客户端  CloseableHttpClient client = HttpClientBuilder.create().build();  #2、发送基本的GET请求  instance.execute(new HttpGet(“http://www.baidu.com”));  #3、获取HTTP响应的状态码  String url = “http://www.baidu.com”;  CloseableHttpResponse response = instance.execute(new HttpGet(url));  assertThat(response.getStatusLine().getStatusCode(), equalTo(200));  #4、获取响应的媒体类型  String url = “http://www.baidu.com”;  CloseableHttpResponse response = instance.execute(new HttpGet(url));  String contentMimeType = ContentType.getOrDefault(response.getEntity()).getMimeType();  assertThat(contentMimeType, equalTo(ContentType.TEXT\_HTML.getMimeType()));  #5、获取响应的BODY部分  String url = “http://www.baidu.com”;  CloseableHttpResponse response = instance.execute(new HttpGet(url));  String bodyAsString = EntityUtils.toString(response.getEntity());  assertThat(bodyAsString, notNullValue());  #6、配置请求的超时设置  @Test(expected=SocketTimeoutException.class)  public void givenLowTimeout\_whenExecutingRequestWithTimeout\_thenException() throws ClientProtocolException, IOException{  RequestConfig requestConfig = RequestConfig.custom()  .setConnectionRequestTimeout(50).setConnectTimeout(50)  .setSocketTimeout(50).build();  HttpGet request = new HttpGet(SAMPLE\_URL);  request.setConfig(requestConfig);  instance.execute(request);  }  #7、发送POST请求  instance.execute(new HttpPost(SAMPLE\_URL));  #8、为HTTP请求配置重定向  CloseableHttpClient instance = HttpClientBuilder.create().disableRedirectHandling().build();  CloseableHttpResponse response = instance.execute(new HttpGet(SAMPLE\_URL));  assertThat(reponse.getStatusLine().getStatusCode(), equalTo(301));  #9、配置请求的HEADER部分  HttpGet request = new HttpGet(SAMPLE\_URL);  request.addHeader(HttpHeaders.ACCEPT, “application/xml”);  response = instance.execute(request);  #10、获取响应的HEADER部分  CloseableHttpResponse response = instance.execute(new HttpGet(SAMPLE\_URL));  Header[] headers = response.getHeaders(HttpHeaders.CONTENT\_TYPE);  assertThat(headers, not(emptyArray()));  #11、关闭或释放资源  response = instance.execute(new HttpGet(SAMPLE\_URL));  try{  HttpEntity entity = response.getEntity();  if(entity!=null){  InputStream instream = entity.getContent();  instream.close();  }  } finally{  response.close();  } |

# 使用实例

## post，get请求示例

|  |
| --- |
| **public** **class** HttpClientTest {  **public** **static** **void** main(String[] args) **throws** Exception {  *get*();  }  /\*\*  \* post方式提交json代码  \*  \* **@throws** Exception  \*/  **public** **static** **void** postJson() **throws** Exception {  // 创建默认的httpClient实例.  CloseableHttpClient httpclient = **null**;  // 接收响应结果  CloseableHttpResponse response = **null**;  **try** {  // 创建httppost  httpclient = HttpClients.*createDefault*();  String url = "http://192.168.16.36:8081/goSearch/gosuncn/deleteDocs.htm";  HttpPost httpPost = **new** HttpPost(url);  httpPost.addHeader(HTTP.***CONTENT\_TYPE***, "application/x-www-form-urlencoded");  // 参数  String json = "{'ids':['html1','html2']}";  StringEntity se = **new** StringEntity(json);  se.setContentEncoding("UTF-8");  se.setContentType("application/json");// 发送json需要设置contentType  httpPost.setEntity(se);  response = httpclient.execute(httpPost);  // 解析返结果  HttpEntity entity = response.getEntity();  **if** (entity != **null**) {  String resStr = EntityUtils.*toString*(entity, "UTF-8");  System.***out***.println(resStr);  }  } **catch** (Exception e) {  **throw** e;  } **finally** {  httpclient.close();  response.close();  }  }  /\*\*  \* post方式提交表单（模拟用户登录请求）  \*  \* **@throws** Exception  \*/  **public** **static** **void** postForm() **throws** Exception {  // 创建默认的httpClient实例.  CloseableHttpClient httpclient = **null**;  // 发送请求  CloseableHttpResponse response = **null**;  **try** {  httpclient = HttpClients.*createDefault*();  // 创建httppost  String url = "http://localhost:8080/search/ajx/user.htm";  HttpPost httppost = **new** HttpPost(url);  // 创建参数队列  List<NameValuePair> formparams = **new** ArrayList<NameValuePair>();  formparams.add(**new** BasicNameValuePair("username", "admin"));  formparams.add(**new** BasicNameValuePair("password", "123456"));  // 参数转码  UrlEncodedFormEntity uefEntity = **new** UrlEncodedFormEntity(formparams, "UTF-8");  httppost.setEntity(uefEntity);  response = httpclient.execute(httppost);  HttpEntity entity = response.getEntity();  **if** (entity != **null**) {  System.***out***.println(EntityUtils.*toString*(entity, "UTF-8"));  }  // 释放连接  } **catch** (Exception e) {  **throw** e;  } **finally** {  httpclient.close();  response.close();  }  }  /\*\*  \* 发送 get请求  \*  \* **@throws** Exception  \*/  **public** **static** **void** get() **throws** Exception {  CloseableHttpClient httpclient = **null**;  CloseableHttpResponse response = **null**;  **try** {  httpclient = HttpClients.*createDefault*();  // 创建httpget.  HttpGet httpget = **new** HttpGet("http://www.baidu.com/");  // 执行get请求.  response = httpclient.execute(httpget);  // 获取响应实体  HttpEntity entity = response.getEntity();  // 打印响应状态  System.***out***.println(response.getStatusLine().getStatusCode());  **if** (entity != **null**) {  // 打印响应内容  System.***out***.println("Response content: " + EntityUtils.*toString*(entity));  }  } **catch** (Exception e) {  **throw** e;  } **finally** {  httpclient.close();  response.close();  }  }  } |