CloseableHttpAsyncClient，偶尔会报下面的错误

1、java.io.IOException: Connection reset by peer

1）服务器的并发连接数超过了其承载量，服务器会将其中一些连接关闭

网络断了一下，客户段不知道，继续发信息，就会报错。 连接被对方重置了。

发送数据的时候，连接被断掉了，汇报这个错。

可能原因：

2、Tomcat服务器在接受用户请求的时候，有其自身的处理能力，线程、服务器等各个资源限制，超出Tomcat承载范围的请求，就会被tomcat停掉，也可能产生该错误。

2、org.apache.http.ConnectionClosedException: Connection closed

网络问题

请求还没结束，获取返回数据，连接被断掉了，汇报这个错

[**HttpAsyncClient的连接池使用**](https://www.cnblogs.com/zemliu/p/3719292.html)

https://www.cnblogs.com/zemliu/p/3719292.html

讲的比较清楚

3、post请求——表单方式、json方式

|  |
| --- |
| public static String httpPost(String host, Map<String, Object> param)throws Exception {  HttpPost httpPost = new HttpPost(host);   //表单方式  List<NameValuePair> nameValuePairs = new ArrayList<NameValuePair>();  for (Map.Entry<String, Object> entry : param.entrySet()) {  nameValuePairs.add(new BasicNameValuePair(entry.getKey(), entry.getValue().toString()));  }  UrlEncodedFormEntity urlEncodedFormEntity = new UrlEncodedFormEntity(nameValuePairs, UtilConstant.*httpCharset*);  httpPost.setEntity(urlEncodedFormEntity);   Future<HttpResponse> future = *httpclient*.execute(httpPost, null);  String response = EntityUtils.*toString*(future.get().getEntity(), UtilConstant.*httpCharset*);  return response; }  public static String httpPost(String host, String param)throws Exception {  HttpPost httpPost = new HttpPost(host);  //construct request header  httpPost.setHeader("Content-type", "application/json; charset=UTF-8");  StringEntity entity = new StringEntity(param);  httpPost.setEntity(entity);  Future<HttpResponse> future = *httpclient*.execute(httpPost, null);  String response = EntityUtils.*toString*(future.get().getEntity(), UtilConstant.*httpCharset*);  return response; } |

通信过程中网络断掉会报错：

|  |
| --- |
| Caused by: java.io.IOException: 远程主机强迫关闭了一个现有的连接。  at sun.nio.ch.SocketDispatcher.read0(Native Method)  at sun.nio.ch.SocketDispatcher.read(SocketDispatcher.java:43)  at sun.nio.ch.IOUtil.readIntoNativeBuffer(IOUtil.java:223)  at sun.nio.ch.IOUtil.read(IOUtil.java:197)  at sun.nio.ch.SocketChannelImpl.read(SocketChannelImpl.java:380)  at org.apache.http.impl.nio.reactor.SessionInputBufferImpl.fill(SessionInputBufferImpl.java:204)  at org.apache.http.impl.nio.codecs.AbstractMessageParser.fillBuffer(AbstractMessageParser.java:136)  at org.apache.http.impl.nio.DefaultNHttpClientConnection.consumeInput(DefaultNHttpClientConnection.java:241)  at org.apache.http.impl.nio.client.InternalIODispatch.onInputReady(InternalIODispatch.java:81)  at org.apache.http.impl.nio.client.InternalIODispatch.onInputReady(InternalIODispatch.java:39)  at org.apache.http.impl.nio.reactor.AbstractIODispatch.inputReady(AbstractIODispatch.java:114)  at org.apache.http.impl.nio.reactor.BaseIOReactor.readable(BaseIOReactor.java:162)  at org.apache.http.impl.nio.reactor.AbstractIOReactor.processEvent(AbstractIOReactor.java:337)  at org.apache.http.impl.nio.reactor.AbstractIOReactor.processEvents(AbstractIOReactor.java:315)  at org.apache.http.impl.nio.reactor.AbstractIOReactor.execute(AbstractIOReactor.java:276)  at org.apache.http.impl.nio.reactor.BaseIOReactor.execute(BaseIOReactor.java:104)  at org.apache.http.impl.nio.reactor.AbstractMultiworkerIOReactor$Worker.run(AbstractMultiworkerIOReactor.java:588)  at java.lang.Thread.run(Thread.java:748) |

服务端网络断掉，客户端不知道，请求连接报错

|  |
| --- |
|  |

超时异常

|  |
| --- |
| Caused by: java.net.SocketTimeoutException  at org.apache.http.nio.protocol.HttpAsyncRequestExecutor.timeout(HttpAsyncRequestExecutor.java:375)  at org.apache.http.impl.nio.client.InternalIODispatch.onTimeout(InternalIODispatch.java:92)  at org.apache.http.impl.nio.client.InternalIODispatch.onTimeout(InternalIODispatch.java:39)  at org.apache.http.impl.nio.reactor.AbstractIODispatch.timeout(AbstractIODispatch.java:175)  at org.apache.http.impl.nio.reactor.BaseIOReactor.sessionTimedOut(BaseIOReactor.java:263)  at org.apache.http.impl.nio.reactor.AbstractIOReactor.timeoutCheck(AbstractIOReactor.java:492)  at org.apache.http.impl.nio.reactor.BaseIOReactor.validate(BaseIOReactor.java:213)  at org.apache.http.impl.nio.reactor.AbstractIOReactor.execute(AbstractIOReactor.java:280)  at org.apache.http.impl.nio.reactor.BaseIOReactor.execute(BaseIOReactor.java:104)  at org.apache.http.impl.nio.reactor.AbstractMultiworkerIOReactor$Worker.run(AbstractMultiworkerIOReactor.java:588) |

connectionClosedExpception的解释

|  |
| --- |
| It is a classic race condition.  (1) The server apparently has been configured to close connections after 5 seconds of inactivity.  (2) The client polls every < 5 seconds. It leases a persistent connection from the pool which it is still valid. All is well.  (3) The client polls every 5 seconds. It leases a persistent connection from the pool assuming it is still valid. The server closes the connection on its end while the client is still busy executing the request. Message exchange fails with ConnectionClosedException on the client side. Kaboom.  (4) The client polls every > 5 seconds. Persistent connections get closed by the server after 5 seconds of inactivity and gets evicted from the pool of connections on the client side. When the client requests a connection from the pool there is none available. Client opens a new connection. All is well. |