**HttpClient超时设置详解**

2017年11月29日 14:22:02 [追不上老鼠的猫](https://me.csdn.net/u011191463) 阅读数：22202 标签： [httpclient 超时设置](http://so.csdn.net/so/search/s.do?q=httpclient%20%E8%B6%85%E6%97%B6%E8%AE%BE%E7%BD%AE&t=blog)

HttpClient 4.3。HttpClient这货和Lucene一样，每个版本的API都变化很大，这有点让人头疼。就好比创建一个HttpClient对象吧，每一个版本的都不一样，

3.X是这样的

[?](http://www.open-open.com/lib/view/open1383751765321.html)

|  |  |
| --- | --- |
| 1 | HttpClient httpClient=newDefaultHttpClient(); |

4.3是这样的

[?](http://www.open-open.com/lib/view/open1383751765321.html)

|  |  |
| --- | --- |
| 1 | CloseableHttpClient httpClient = HttpClients.createDefault(); |

当然，上面这些变化只不过是一些小变化，大家看看API大家就都会了。

我要讲的是超时设置,HttpClient有三种超时设置，最近比较忙，没时间具体归纳总结，以后再补上，我这里就讲一些最简单最易用的超时设置方法。

这是个3.X的超时设置方法

[?](http://www.open-open.com/lib/view/open1383751765321.html)

|  |  |
| --- | --- |
| 1  2  3 | HttpClient client = newHttpClient();  client.setConnectionTimeout(30000);  client.setTimeout(30000); |

[?](http://www.open-open.com/lib/view/open1383751765321.html)

|  |  |
| --- | --- |
| 1  2 | HttpClient httpClient= newHttpClient();  httpClient.getHttpConnectionManager().getParams().setConnectionTimeout(5000); |

4.X版本的超时设置(4.3后已过时)

[?](http://www.open-open.com/lib/view/open1383751765321.html)

|  |  |
| --- | --- |
| 1  2  3 | HttpClient httpClient=newDefaultHttpClient();  httpClient.getParams().setParameter(CoreConnectionPNames.CONNECTION\_TIMEOUT,2000);//连接时间  httpClient.getParams().setParameter(CoreConnectionPNames.SO\_TIMEOUT,2000);//数据传输时间 |

4.3版本超时设置

[?](http://www.open-open.com/lib/view/open1383751765321.html)

|  |  |
| --- | --- |
| 1  2  3  4  5 | CloseableHttpClient httpClient = HttpClients.createDefault();  HttpGet httpGet=newHttpGet("http://www.baidu.com");//HTTP Get请求(POST雷同)  RequestConfig requestConfig = RequestConfig.custom().setSocketTimeout(2000).setConnectTimeout(2000).build();//设置请求和传输超时时间  httpGet.setConfig(requestConfig);  httpClient.execute(httpGet);//执行请求 |

BTW,4.3版本不设置超时的话，一旦服务器没有响应，等待时间N久(>24小时)。

       HttpClient升级到4.5版本后，API有很多变化，HttpClient 4之后，API一直没有太稳定，我感觉4.5版本抽象后，很多API应该快稳定了。

       使用HttpClient，一般都需要设置连接超时时间和获取数据超时时间。这两个参数很重要，目的是为了防止访问其他http时，由于超时导致自己的应用受影响。

       4.5版本中，这两个参数的设置都抽象到了RequestConfig中，由相应的Builder构建，具体的例子如下：

[?](http://my.oschina.net/wallechen/blog/526642)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | CloseableHttpClient httpclient = HttpClients.createDefault();  HttpGet httpGet = new HttpGet("http://stackoverflow.com/");  RequestConfig requestConfig = RequestConfig.custom()          .setConnectTimeout(5000).setConnectionRequestTimeout(1000)          .setSocketTimeout(5000).build();  httpGet.setConfig(requestConfig);  CloseableHttpResponse response = httpclient.execute(httpGet);  System.out.println("得到的结果:" + response.getStatusLine());//得到请求结果  HttpEntity entity = response.getEntity();//得到请求回来的数据 |

setConnectTimeout：设置连接超时时间，单位毫秒。

setConnectionRequestTimeout：设置从connect Manager(连接池)获取Connection 超时时间，单位毫秒。这个属性是新加的属性，因为目前版本是可以共享连接池的。

setSocketTimeout：请求获取数据的超时时间(即响应时间)，单位毫秒。 如果访问一个接口，多少时间内无法返回数据，就直接放弃此次调用。

HttpClient内部有三个超时时间设置：连接池获取可用连接超时，连接超时，读取数据超时

先看以下HttpClient的初始化代码：

[java] [view plain](http://blog.csdn.net/zhongzh86/article/details/46348933) [copy](http://blog.csdn.net/zhongzh86/article/details/46348933)

1. RequestConfig requestConfig = RequestConfig.custom()
2. .setConnectionRequestTimeout(config.connReqTimeout)   //从连接池中获取连接的超时时间
3. //与服务器连接超时时间：httpclient会创建一个异步线程用以创建socket连接，此处设置该socket的连接超时时间
4. .setConnectTimeout(config.connTimeout)
5. .setSocketTimeout(config.socketTimeout)               //socket读数据超时时间：从服务器获取响应数据的超时时间
6. .build();
7. httpClient = HttpClientBuilder.create()
8. .setMaxConnTotal(config.maxConnTotal)                   //连接池中最大连接数
9. /\*\*
10. \* 分配给同一个route(路由)最大的并发连接数。
11. \* route：运行环境机器 到 目标机器的一条线路。
12. \* 举例来说，我们使用HttpClient的实现来分别请求 www.baidu.com 的资源和 www.bing.com 的资源那么他就会产生两个route。
13. \*/
14. .setMaxConnPerRoute(config.maxConnPerRoute)
15. .setDefaultRequestConfig(requestConfig)
16. .build();

三个超时时间详解：

1.从连接池中获取可用连接超时

HttpClient中的要用连接时尝试从连接池中获取，若是在等待了一定的时间后还没有获取到可用连接（比如连接池中没有空闲连接了）

则会抛出获取连接超时异常。

2.连接目标超时connectionTimeout

指的是连接目标*url*的连接超时时间，即客服端发送请求到与目标*url*建立起连接的最大时间。如果在该时间范围内还没有建立起连接，则就

抛出*connectionTimeOut*异常。*如*测试的时候，将*url*改为一个不存在的*url*：*“http://test.com” ，*超时时间*3000ms*过后，系统报出异常：

*org.apache.commons.httpclient.ConnectTimeoutException:The host did not accept the connection within timeout of 3000 ms*

*3.等待响应超时（读取数据超时）socketTimeout*

   连接上一个url后，获取response的返回等待时间 ，即在与目标url建立连接后，等待放回response的最大时间，在规定时间内没有返回响应的话就抛出SocketTimeout。  
   测试的时候的连接url为我本地开启的一个url，http://localhost:8080/firstTest.htm?method=test，在我这个测试url里，当访问到这个链接时，线程sleep一段时间，来模拟返回response超时。

Java代码

1. @RequestMapping(params = "method=test")

2. public String testMethod(ModelMap model) {

3. try {

4.     Thread.sleep(10000);

5. } catch (InterruptedException e) {

6.     // TODO Auto-generated catch block

7.     e.printStackTrace();

8. }

9.       System.out.println("call testMethod method.");

10.       model.addAttribute("name", "test method");

11. return "test";

12.   }

   将读取response返回超时时间设的时间比那个sleep时间短之后，运行程序给出异常：java.net.SocketTimeoutException:Read timed out

参考：http://blog.csdn.net/waterCabin/article/details/50294787

          http://blog.csdn.net/zhongzh86/article/details/46348933