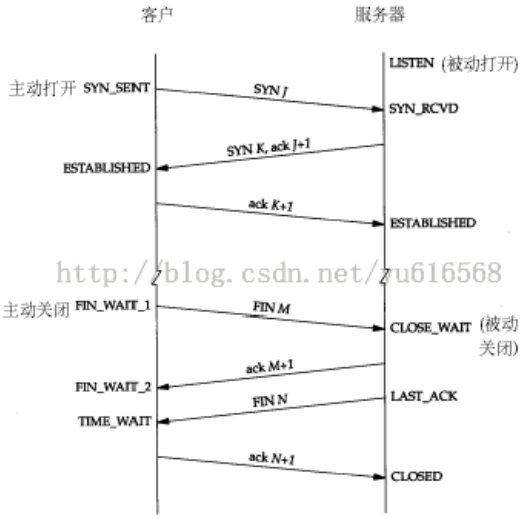
一次服务端大量CLOSE_WAIT问题的解决

缘由

今天在运行服务器的时候发现一个问题，问题的表现是客户端一直在请求，但是返回给客户端的信息是异常，服务端压根没有收到请求，查看了一下配置信息没有错误，首先查看了一下是不是服务器的连接已经满了，打开netstat命令发现服务器的连接有大量的CLOSE_WAIT状态的socket，没怎么遇到这个问题。

分析

开始还真有段懵了，第一反应就是是不是客户端的问题（是不是出问题的第一反应都是别人的问题），但是马上补充了一下socket状态机的知识，发现这个状态是由于客户端关闭了socket连接，发送了FIN报文，服务端也发送了ACK报文，此时客户端处于FIN*WAIT*2状态，服务端处于CLOSE_WAIT状态，如下图：



可以看出，出现问题的原因是由于我这边没有发送第二个FIN报文导致的，分明是我的问题啊，为什么服务器没有发送FIN报文呢？我的服务器使用的是嵌入式的jetty，连接管理应该都是它帮我管理的，重启了一下服务器发现服务器的CLOSE_WAIT开始的时候没有出现，之后逐渐的上升，貌似随着请求的数量逐渐增长的，而我这边的日志也非常奇怪，我会在收到请求的时候打印日志，然后在执行完毕的时候输出一个accesslog信息，发现日志中有入口的请求日志，但是accessLog没有增长，于是单步调试了一下，发现了问题：一个servlet的执行走到主流程就走不下去了，阻塞在数据库访问那一步上，具体表现就是获取不到数据库连接！

查看了一下代码，发现原来是自己创建连接，执行sql，完成之后没有关闭连接，OMG，这么愚蠢的错误，于是在所有的数据库操作的最后加上如下的代码：

```
finally {
    DbUtils.closeQuietly(conn);
}
```

好了，既然问题能够解决了，现在回头来思考一下问题产生的具体步骤：首先，我这边的大部分请求都需要查询数据库，我的数据库连接池设置的最大连接数是100，所以每一个请求创建了一个连接，等到100个请求就把连接池占满了，但是处理servlet的那个线程并没有释放这个连接，于是接下来的请求再去创建数据库连接的时候就会一直阻塞在那里，这里我所用的是DBCP作为连接池的，它的实现好像是使用apache的objectPool来实现的，如果没有可用的连接对象会导致线程等待，好了，servlet由于得不到数据库连接而阻塞了，这个客户端的请求就一直等待，客户端使用httpclient设置了5s的请求超时时间，那么超时之后就会抛出异常，关闭连接，关闭连接导致客户端发送了FIN报文，我这边的TCP/IP返回了ACK报文，但是由于处理请求的线程还处于阻塞的状态，所以当前的连接状态时CLOSE_WAIT。

警示：

- 代码一定要规范，尤其是在写一些关于自愿申请的部分，一定要在写函数之前写上注释告诉自己别忘了释放资源。
- 数据库连接和访问要设置超时时间，这点避免阻塞。
- 服务器的线程数也需要设置，使得问题尽可能的出现。