可变状态变量在并发中解决方案:

1. 不在线程之间共享该状态变量
2. 将状态变量修改为不可变
3. 在访问状态变量时使用同步

JAVA的内置锁是互斥锁，最多只有一个线程持有该锁

当执行时间较长的计算或是可能无法完成的操作如IO，一定不要加锁。

非volatile的64位变量(double/long)的读写操作分为两个32位操作，所以也不是原子操作，即非线程安全

加锁的含义不仅仅局限于互斥行为，还包括内存可见性，为了确保所有线程都能看到共享变量的最新值，所以执行读操作或者写操作的线程都必须在同一个锁上同步。

volatile是削弱版的同步机制，被其修饰的变量会在所有线程可见，并总是读到最新值。

但是volatile无法保证类似count++操作的原子性，所以syncronise加锁机制即可以保证原子性和可见性，但volatile只能保证可见性

线程封闭，即使用单线程的形式访问数据。常见的JDBC Connection和ThreadLocal就是线程封闭的。

在孔浩的项目里，Contoller如果需要向Service传递大量参数时，会造成性能上的影响。所以使用了ThreadLocal来存储这些通用参数，从这个线程中直接读取参数就会效率更高。

ThreadLocal为每个线程提供了一份变量副本，而不会和其他线程的副本冲突。JDBC connection可以被多线程共享，所以在不加锁的情况下是非线程安全的，如A线程事务没结束，B线程获得了该connection然后commit了，数据就会发生混乱。通过ThreadLocal<connection>每个线程都获得自己的connection副本，所以之间不会冲突，也就线程安全了。