[https://mp.weixin.qq.com/s?\_\_biz=MzA5MTkxMDQ4MQ==&mid=2648933186&idx=1&sn=079567e8799e43cb734b833c44055c01&chksm=88621b7cbf15926aace88777445822314d6eed2c1f5559b36cb6a6e181f0e543ee14d832ebc2&token=1963100670&lang=zh\_CN&scene=21#wechat\_redirect](https://mp.weixin.qq.com/s?__biz=MzA5MTkxMDQ4MQ==&mid=2648933186&idx=1&sn=079567e8799e43cb734b833c44055c01&chksm=88621b7cbf15926aace88777445822314d6eed2c1f5559b36cb6a6e181f0e543ee14d832ebc2&token=1963100670&lang=zh_CN&scene=21" \l "wechat_redirect)

# ThreadLocal、InheritableThreadLocal

## **本文内容**

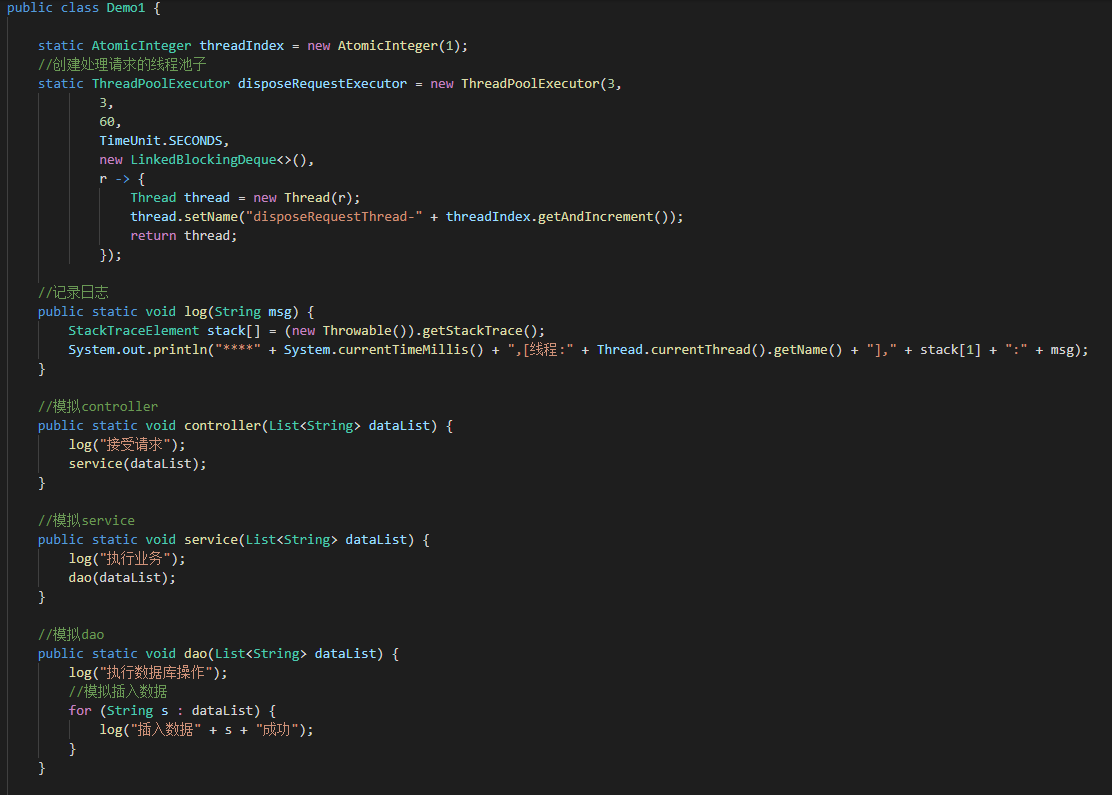
* 需要解决的问题
* 介绍ThreadLocal
* 介绍InheritableThreadLocal

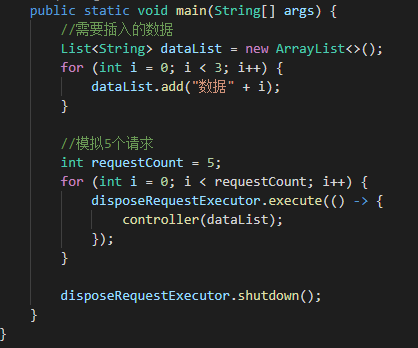
## **需要解决的问题**

我们还是以解决问题的方式来引出ThreadLocal、InheritableThreadLocal，这样印象会深刻一些。

目前java开发web系统一般有3层，controller、service、dao，请求到达controller，controller调用service，service调用dao，然后进行处理。

我们写一个简单的例子，有3个方法分别模拟controller、service、dao。代码如下：





运行结果：

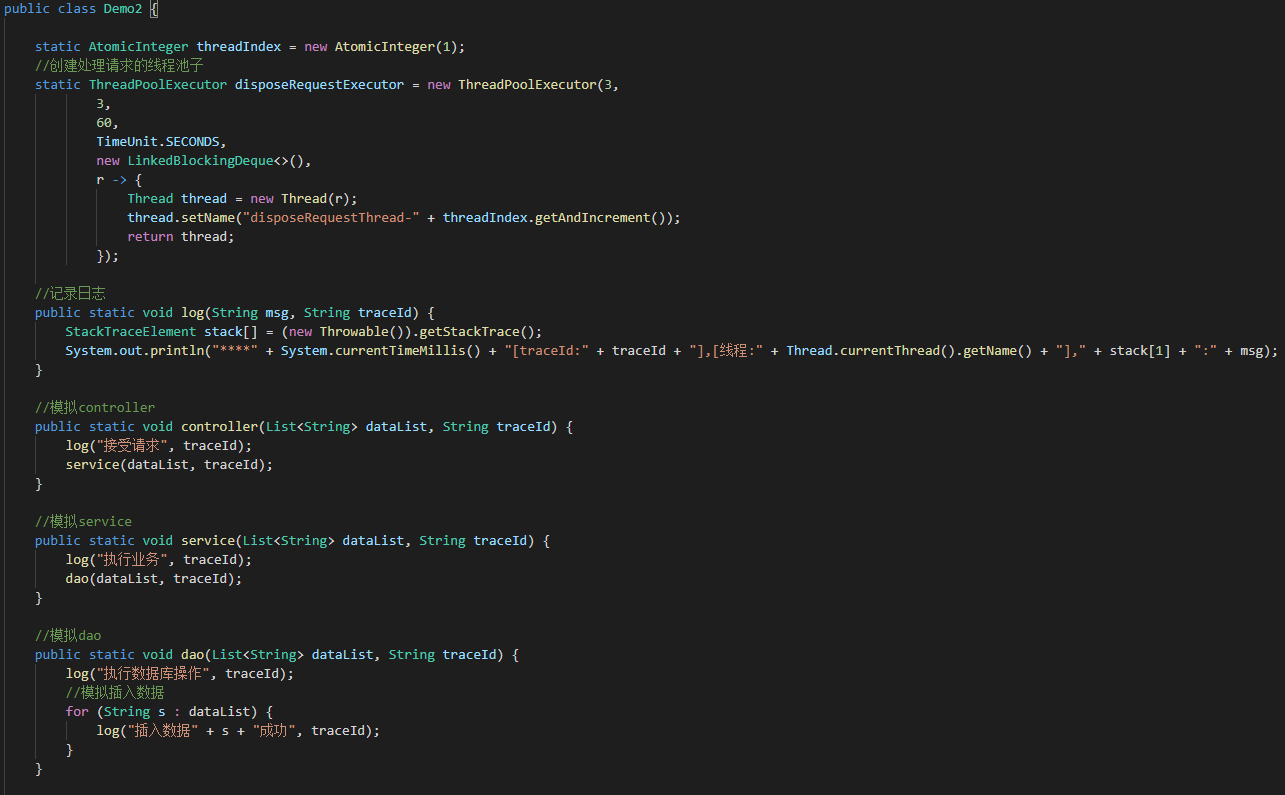


代码中调用controller、service、dao 3个方法时，来模拟处理一个请求。main方法中循环了5次模拟发起5次请求，然后交给线程池去处理请求，dao中模拟循环插入传入的dataList数据。

**问题来了：开发者想看一下哪些地方耗时比较多，想通过日志来分析耗时情况，想追踪某个请求的完整日志，怎么搞？**

上面的请求采用线程池的方式处理的，多个请求可能会被一个线程处理，通过日志很难看出那些日志是同一个请求，我们能不能给请求加一个唯一标志，日志中输出这个唯一标志，当然可以。

如果我们的代码就只有上面示例这么简单，我想还是很容易的，上面就3个方法，给每个方法加个traceId参数，log方法也加个traceId参数，就解决了，代码如下：





输出：



上面我们通过修改代码的方式，把问题解决了，但前提是你们的系统都像上面这么简单，功能很少，需要改的代码很少，可以这么去改。但事与愿违，我们的系统一般功能都是比较多的，如果我们都一个个去改，岂不是要疯掉，改代码还涉及到重新测试，风险也不可控。那有什么好办法么？

## **ThreadLocal**

还是拿上面的问题，我们来分析一下，每个请求都是由一个线程处理的，线程就相当于一个人一样，每个请求相当于一个任务，任务来了，人来处理，处理完毕之后，再处理下一个请求任务。人身上是不是有很多口袋，人刚开始准备处理任务的时候，我们把任务的编号放在处理者的口袋中，然后处理中一路携带着，处理过程中如果需要用到这个编号，直接从口袋中获取就可以了。那么刚好java中线程设计的时候也考虑到了这些问题，Thread对象中就有很多口袋，用来放东西。Thread类中有这么一个变量：

ThreadLocal.ThreadLocalMap threadLocals = null;

这个就是用来操作Thread中所有口袋的东西，ThreadLocalMap源码中有一个数组（有兴趣的可以去看一下源码），对应处理着身上很多口袋一样，数组中的每个元素对应一个口袋。

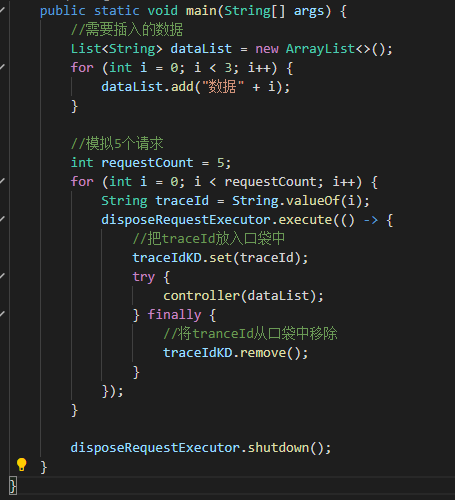
如何来操作Thread中的这些口袋呢，java为我们提供了一个类ThreadLocal，ThreadLocal对象用来操作Thread中的某一个口袋，可以向这个口袋中放东西、获取里面的东西、清除里面的东西，这个口袋一次性只能放一个东西，重复放东西会将里面已经存在的东西覆盖掉。

常用的3个方法：

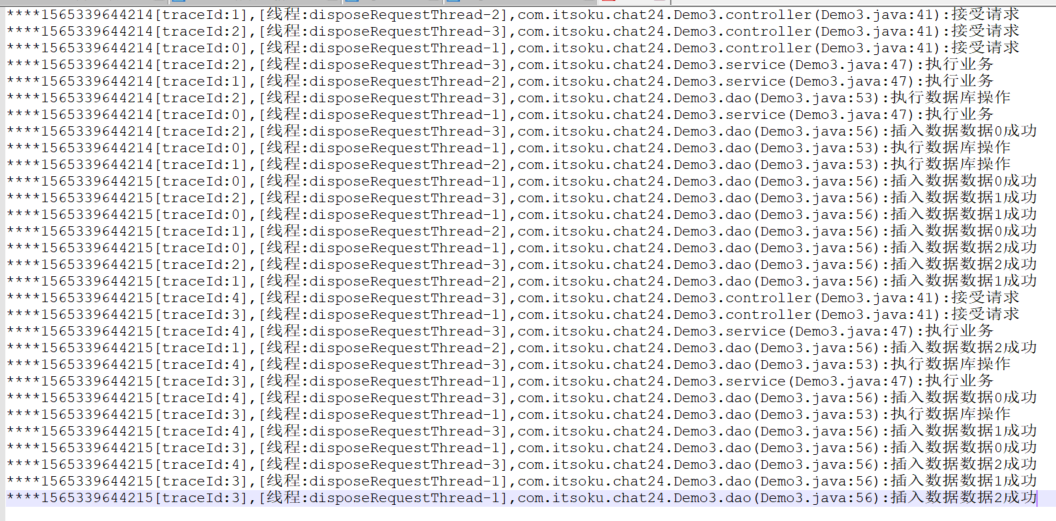
//向Thread中某个口袋中放东西  
public void set(T value);  
//获取这个口袋中目前放的东西  
public T get();  
//清空这个口袋中放的东西  
public void remove()

我们使用ThreadLocal来改造一下上面的代码，如下：





输出：



可以看出输出和刚才使用traceId参数的方式结果一致，但是却简单了很多。不用去修改controller、service、dao代码了，风险也减少了很多。

代码中创建了一个ThreadLocal traceIdKD，这个对象用来操作Thread中一个口袋，用这个口袋来存放tranceId。在main方法中通过traceIdKD.set(traceId)方法将traceId放入口袋，log方法中通traceIdKD.get()获取口袋中的traceId，最后任务处理完之后，main中的finally中调用traceIdKD.remove();将口袋中的traceId清除。

**ThreadLocal的官方API解释为：**

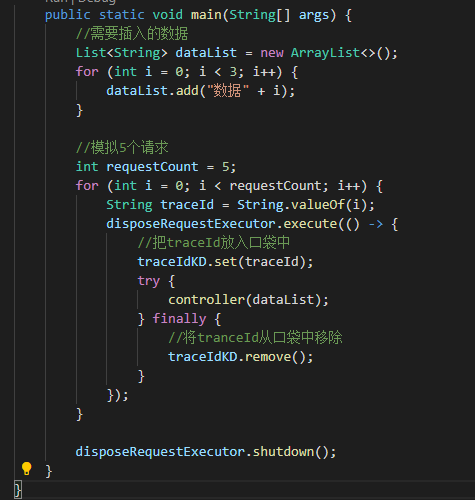
“该类提供了线程局部 (thread-local) 变量。这些变量不同于它们的普通对应物，因为访问某个变量（通过其 get 或 set 方法）的每个线程都有自己的局部变量，它独立于变量的初始化副本。ThreadLocal 实例通常是类中的 private static 字段，它们希望将状态与某一个线程（例如，用户 ID 或事务 ID）相关联。”

## **InheritableThreadLocal**

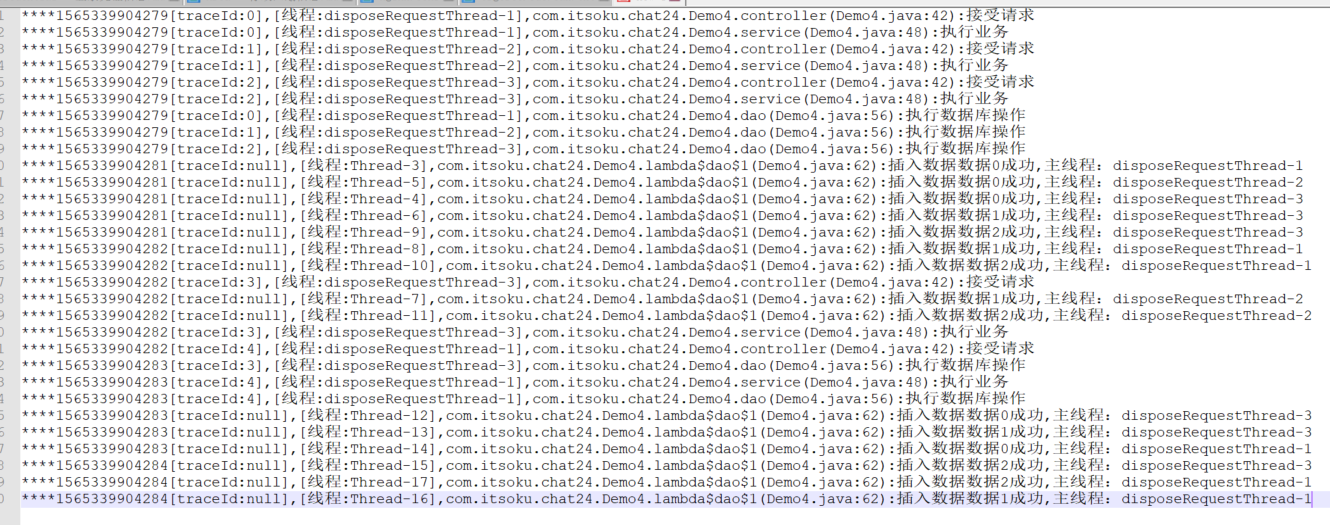
继续上面的实例，dao中循环处理dataList的内容，假如dataList处理比较耗时，我们想加快处理速度有什么办法么？大家已经想到了，用多线程并行处理dataList，那么我们把代码改一下，如下：







输出：



看一下上面的输出，有些traceId为null，这是为什么呢？这是因为dao中为了提升处理速度，创建了子线程来并行处理，子线程调用log的时候，去自己的存放traceId的口袋中拿去东西，肯定是空的了。

那有什么办法么？可不可以这样？

父线程相当于主管，子线程相当于干活的小弟，主管让小弟们干活的时候，将自己兜里面的东西复制一份给小弟们使用，主管兜里面可能有很多牛逼的工具，为了提升小弟们的工作效率，给小弟们都复制一个，丢到小弟们的兜里，然后小弟就可以从自己的兜里拿去这些东西使用了，也可以清空自己兜里面的东西。

Thread对象中有个inheritableThreadLocals变量，代码如下：

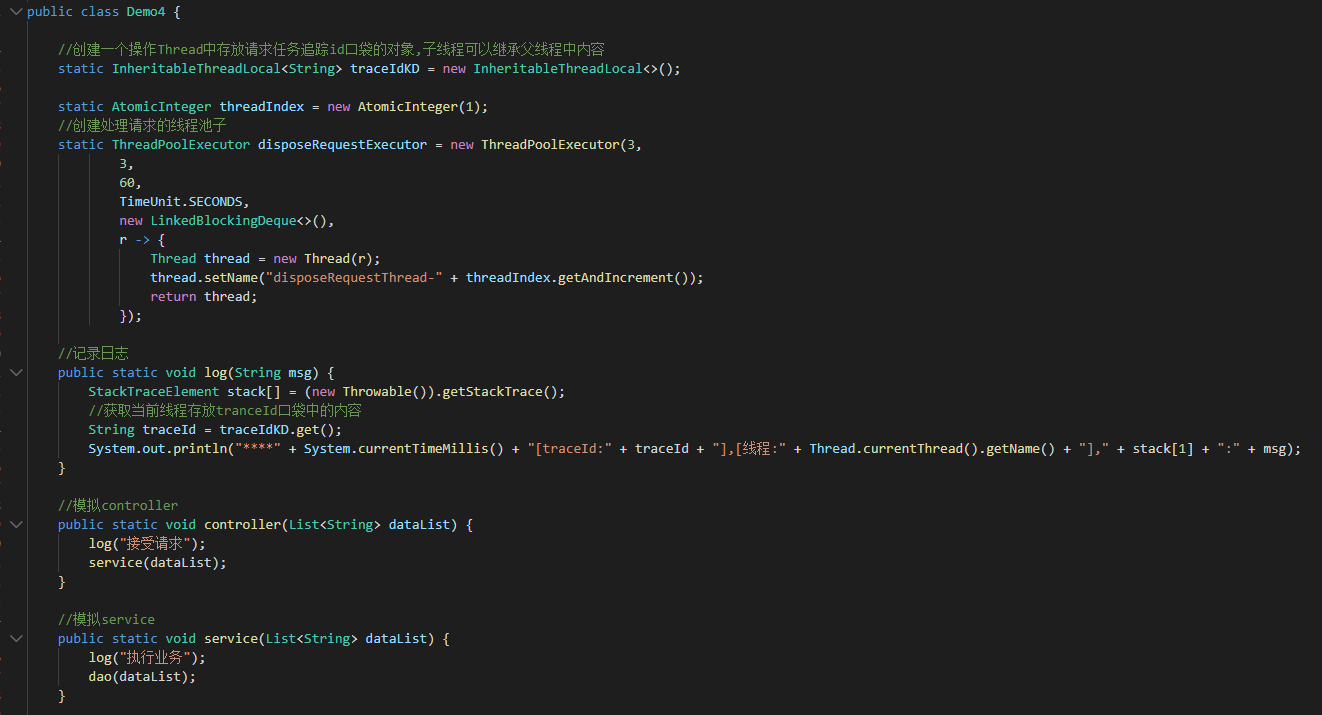
ThreadLocal.ThreadLocalMap inheritableThreadLocals = null;

inheritableThreadLocals相当于线程中另外一种兜，这种兜有什么特征呢，当创建子线程的时候，子线程会将父线程这种类型兜的东西全部复制一份放到自己的inheritableThreadLocals兜中，使用InheritableThreadLocal对象可以操作线程中的inheritableThreadLocals兜。

InheritableThreadLocal常用的方法也有3个：

//向Thread中某个口袋中放东西  
public void set(T value);  
//获取这个口袋中目前放的东西  
public T get();  
//清空这个口袋中放的东西  
public void remove()

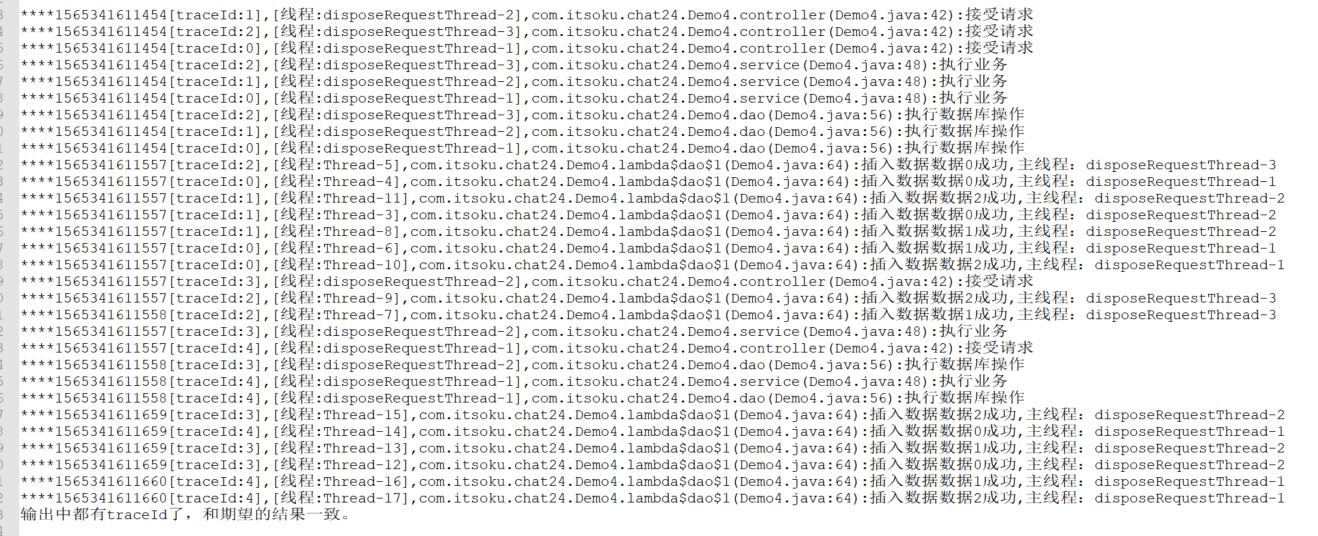
使用InheritableThreadLocal解决上面子线程中无法输出traceId的问题，只需要将上一个示例代码中的ThreadLocal替换成InheritableThreadLocal即可，代码如下：







输出：



输出中都有traceId了，和期望的结果一致。

希望通过这篇文章可以学会使用InheritableThreadLocal和InheritableThreadLocal。