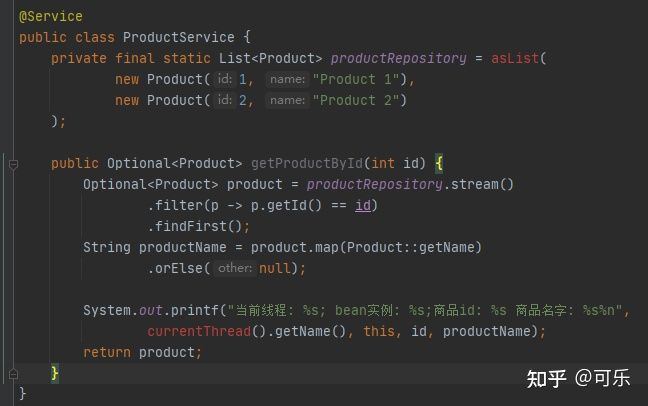
弄明白下面问题，有助于理解JVM的部分概念。

在实际项目中Controller、Service、Dao层的类都是单例的，那么在并发场景下如何保证每个请求之间数据的安全呢，重点就是在于JVM的堆内容。

1.概述  
了解使用单例范围创建的 Spring bean 如何在幕后工作以服务多个并发请求。此外，将了解 Java 如何将 bean 实例存储在内存中以及它如何处理对它们的并发访问。

2. Spring Beans 和 Java 堆内存  
Java 堆是应用程序中所有正在运行的线程都可以访问的全局共享内存。当 Spring 容器创建具有单例范围的 bean 时，该 bean 存储在堆中。这样，所有并发线程都能够指向同一个 bean 实例。

3. 如何处理并发请求  
举个例子，看一个 Spring 应用程序，它有一个名为ProductService的单例 bean ：



这个 bean 有一个方法getProductById()，它将产品数据返回给它的调用者。此外，此 bean 返回的数据在端点 \*/product/{id}\*上暴露给客户端。  
接下来，探索当同时调用\*/product/{id}时在运行时会发生什么。具体来说，第一个线程将调用端点/product/1\*，第二个线程将调用\*/product/2\*。  
Spring 为每个请求创建一个不同的线程。正如下面的控制台输出中看到的，两个线程都使用相同的ProductService实例来返回产品数据：

IMG_257

Spring 可以在多个线程中使用同一个 bean 实例，首先是因为对于每个线程，Java 都会创建一个私有堆栈内存。堆栈内存负责存储线程执行期间方法内部使用的局部变量的状态。这样，Java 确保并行执行的线程不会覆盖彼此的变量。  
其次，由于ProductService bean 在堆级别没有设置任何限制或锁定，因此每个线程的程序计数器都能够指向堆内存中 bean 实例的相同引用。因此，两个线程可以同时执行getProdcutById()方法。

接下来，了解为什么单例 bean 无状态是至关重要的。  
**无状态类：类中没有状态信息，一般是无成员变量或成员变量的值是不变的。**  
4. 无状态单例 Bean 与有状态单例 Bean  
要了解为什么无状态单例 bean 很重要，看看使用有状态单例 bean 的副作用是什么。  
假设将productName变量移至类级别（即成员变量）：



现在，再次运行服务并查看输出

IMG_259

对productId 1 的调用显示的是productName “Product 2”而不是“Product 1”。发生这种情况是因为ProductService是有状态的，并且与所有正在运行的线程共享相同的productName变量。

为了避免这样的不良副作用，保持单例 bean 无状态至关重要。