作业帮面经

作业帮一面(35分钟)

题目列表

1. hashMap底层实现
2. 深拷贝、浅拷贝
3. synchronized底层原理
4. synchronized的四种状态，所升级
5. volatile作用
6. lock和synchronized的区别
7. 8锁中的2个场景
8. try catch finnal{里面不会影响}
9. 线程池的核心参数，线程池的变化
10. spring事务失效的场景
11. spring boot `@Conditional`条件注解
12. linux命令，如何统计日志中的接口调用次数？awk
13. mybatis的dao和xml映射关系
14. mybatis的延迟加载
15. mybatis的mapper支持重载吗?
16. mybatis里面的标签都有哪些，除了crud，
    还有choose、when、otherwise、where、set、foreach

17. mysql事务隔离级别
18. mysql如何解决幻读和脏读
19. mysql可重复读如何实现的？快照读和当前读没有解释清楚。

问题剖析

1. 深拷贝、浅拷贝

- **浅拷贝**（shallowCopy）只是增加了一个指针指向已存在的内存地址。
- **深拷贝**（deepCopy）是增加了一个指针并且申请了一个新的内存，使这个增加的指针指向这个新的内存。

使用深拷贝的情况下，释放内存的时候不会因为出现浅拷贝时释放同一个内存的错误。

> 实现对象拷贝的类，必须实现Cloneable接口，并覆写clone()方法。
> 慎用 Object 的 clone 方法来拷贝对象
> 对象的 clone 方法默认是浅拷贝，若想实现深拷贝需要重写 clone 方法实现属性对象的拷贝。

String类型非常特殊，String类型的数据是存放在常量池中的，也就是无法修改的！也就是说，当我将name属性从“摇头耶稣”改为“大傻子"后，并不是修改了这个数据的值，而是把这个数据的引用从指向”摇头耶稣“这个常量改为了指向”大傻子“这个常量。在这种情况下，另一个对象的name属性值仍然指向”摇头耶稣“不会受到影响。

2. synchronized底层原理

每个对象都有一个内置的monitor对象，比如一个对象实例有一个monitor，一个类的Class对象也有一个monitor，如果要对这个对象加锁，必须获取这个对象的monitor锁。

monitor里面有一个计数器，从0开始，如果一个线程要获取monitor的锁，就看看他的计数器是不是0，如果是0的话，那么说明没人加锁， 他就可以获取锁了。然后对计数器加1，

当程序进入synchronized修饰的代码范围，JVM底层有一个 monitorenter 的指令，当程序出了synchronized修饰的代码范围，就会有一个 monitorexit指令，此时获取锁的线程会对这个对象的monitor的计数器减1，当计数器减到0，代表释放锁。

> JVM为每一个类或实例维护了一个监视器monitor，当出现synchronized关键字时monitor会做出如下判断
> 1. 判断monitor是否已有所有者，如果没有，则允许线程进入，并把计数器从0变为1
> 2. 如果monitor已经拥有所有者，则当前线程进入等待状态，并把计数器加1

> 3. 当线程退出时，计数器减1。当计数器为0时，monitor失去所有者

3. synchronized的四种状态，锁升级

- **无锁**：对象刚new出来的时候
- **偏向锁**：对象只有一个线程获取锁的时候
- **轻量锁**：多线程交替执行
- **重量锁**：多个线程就会发生竞争膨胀为重量锁

4. spring事务失效的场景

1）没有被 Spring 管理，比如没有加@service注解注入spring容器。

2）方法不是 public的，`@Transactional` 只能用于 public 的方法上，否则事务不生效。

3）发生了自身调用，就调该类自己的方法，而没有经过 Spring 的代理类，默认只有在外部调用事务才会生效，比如一个类中一个没有添加事务注解的方法，去调用一个加了事务注解的方法，事务不生效。

4）数据源没有配置事务管理器。

5）异常被吃了，自己在事务代码中捕获了异常没有抛出来。事务不生效。

6）异常类型错误或格式配置错误，比如抛的是RuntimeException，然后被捕获了自己重新抛了一个Exception。

以上几种情况，发生最多就是自身调用、异常被吃、异常抛出类型不对这三个！

5. spring boot @conditional 条件注解

主要作用就是判断条件是否满足，从而决定是否初始化并向容器注册Bean，

> 通过`@Conditional`注解配合`Condition`接口，来决定给一个bean是否创建和注册到Spring容器中，从而实现有选择的加载bean。

这样做的目的是什么呢?

- 当有多个同名bean时，怎么抉择的问题。
- 解决某些bean的创建有其他依赖条件的情况。

6. mybatis的dao和xml映射关系，Dao里面的方法支持重载吗?

Mybatis在初始化SqlSessionFactoryBean的时候，找到mapperLocations路径去解析里面所有的XML文件

Dao接口并没有实现类，那么，我们在调用它的时候。比如你的项目是基于SpringBoot的，那么肯定也见过这种：`@MapperScan("com.xxx.dao")`；它的作用是将包路径下的所有类注册到Spring Bean中，并且将它们的beanClass设置为MapperFactoryBean。MapperFactoryBean实现了FactoryBean接口，俗称工厂Bean。那么，当我们通过@Autowired注入这个Dao接口的时候，返回的对象就是MapperFactoryBean这个工厂Bean中的getObject()方法对象。

然后这个getObject()方法通过**JDK动态代理**，返回了一个Dao接口的代理对象，这个代理对象的处理器是MapperProxy对象。所有，我们通过@Autowired注入Dao接口的时候，注入的就是这个代理对象，我们调用到Dao接口的方法时，则会调用到MapperProxy对象的invoke方法。

> 只要你配置了MapperScan，它就会去扫描，然后生成代理。

当我们调用Dao接口方法的时候，实际调用到代理对象的invoke方法。 在这里，实际上调用的就是SqlSession里面的东西了。

Dao接口即Mapper接口。接口的全限名（命名空间）就是映射文件中的namespace的值，用于绑定Dao接口；接口的方法名就是映射文件中Mapper的具体sql标签中的id；接口方法内的参数就是传递给sql的参数。

在Mybatis中，每一个 <select>、<insert>、<update>、<delete>标签，都会被解析为一个MapperStatement对象，用于描述一条SQL语句。Mapper接口是没有实现类的，当调用接口方法时，由接口全限名+方法名拼接字符串作为key值，可唯一定位一个MapperStatement。

Mapper接口里的方法，是不能重载的，因为是使用 全限名+方法名 的保存和寻找策略。