JAVA认证辅导:Java事务开发常见问题Java认证考试 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/594/2021_2022_JAVA_E8_AE_A4_E8_AF_81_c104_594703.htm 一、了解事务源 我在面试中经常会问到这样的一个问题，假如有一个全局变量，在一个事务中修改了这个变量的值，而后这个事务因为别的原因回滚了，那这个变量的值会回滚到更改之前的值么? 其实事务只能对它所管理的资源进行提交和回滚，这些资源就是事务源，它通常包括数据库连接资源，JMS队列资源等。事务的ACID(原子性，一致性，隔离性，持久性)属性也是针对它所管理的资源而言的。前面问题中的一个全局变量，可以说是内存中的一块存储空间，那么内存中的数据如何能具备事务属性中的持久性呢?很显然它不在事务的管辖范围之内，也就不会跟着事务的回滚而回滚了。 二、何时回滚事务 在JDBC事务和EJB的Bean管理事务中，我们通常会按下面这种方式控制事务的回滚。在出现某种的异常情况下，我们可以控制让事务回滚，当然也可以提交这个事务。 Connection cn = ... cn.setAutoCommit(false). Statement stmt = cn.createStatement(). try{ stmt.executeUpdate("0update Order..."). cn.commit(). }catch(Exception e) { cn.rollback(). //出现异常，回滚当前事务 }finally{ stmt.close(). cn.close(). } 但是对于EJB的容器管理事务或者Spring的声明式事务，就不大一样了。例如： Connection cn = ... cn.setAutoCommit(false). Statement stmt = cn.createStatement(). try{ stmt.executeUpdate("0update Order..."). cn.commit(). }catch(Exception e) { cn.rollback(). //出现异常，回滚当前事务 }finally{ stmt.close(). cn.close(). } 我们只告诉EJB容

器这个方法需要事务控制，容器会在方法的开始处启动一个事务，在方法返回之前提交这个事务。那如果中间处理过程中出现异常该怎么办?默认情况下只有在RuntimeException和标注为ApplicationException的异常发成时会回滚事务，而在其他的情况下，事务都会提交。也就是说，在异常发生之前所有的操作都会被提交。这就有可能会出现部分提交的问题。有时为了确保一个方法所有的操作都被提交或者回滚，通常会这样做：
@TransactionAttribute(TransactionAttributeType.Required) public void 0updateOrder(Order order) { try{ ... }catch(Exception e) { throw new EJBException(e). } } 在有异常发生时，捕获这个异常，并把它包装成一个EJBException，并重新抛出它。因为EJBException继承了RuntimeException，所以这里抛出一个EJBException告诉EJB容器回滚当前的事务。 另外需要注意的是另外一个方法：setRollbackOnly()，它与EJBException不同的是，它仅仅标记当前事务需要回滚，在方法执行完成之后容器会检查它并回滚事务，它并不抛出任何异常。 相比较EJB，Spring的声明式事务好像控制的更细致一些。来看下面的例子。 @Transactional(rollbackFor={Exception.class}) public void 0updateOrder(Order order) { ... } 我们可以告诉Spring哪种类型的异常需要回滚，当然默认的还是只有在发生RuntimeException时事务会回滚。如果大家也在使用这种事务控制方式的话，还是主动告诉容器何时回滚，何时提交事务吧，采用默认值并不是一个好的办法。 三、事务的范围 以前的一个项目是直接使用JDBC来访问数据库的，后来决定重构到Hibernate和Spring上去，事务控制也由原来的本地JDBC

事务转换为Spring的声明式事务。好像一切都很顺利，直到交给性能测试人员。 更多优质资料尽在百考试题论坛 百考试题在线题库 linux认证更多详细资料 造成性能低下的原因也很明显，就是由于使用了声明式事务，事务的控制范围是从一个方法的开始到它的结束。如果这个方法很长，而只需要确保其中的某几条数据库的操作语句在一个事务中执行，这样，本来只需要一个很短的事务，结果却使用了一个很长的事务。事务一旦过长，它就会影响别的进程或线程来操作同一个事务源。合理的方法应该是在确实需要事务的时候才开始一个事务，而且要及时提交或回滚它，以释放对事务源的访问，这也就是EJB中，使用 BMP(Bean管理事务)相对与CMP(容器管理事务)的好处。如果确定需要在Spring中使用类似BMP的做法，也就是说开发人员自己控制事务，可以通过Spring容器注入Spring的TransactionManager来实现，当然这也就违背了“ 无浸入 ”的原则，需要大家在实际中权衡。 另外Spring的事务标注还给我们提供了一些其他的属性，比如说readOnly，isolation和timeout等。当一个方法需要添加事务控制，而且这个方法只做查询操作的时候，千万别忘记标记“ readOnly = true ”，这样这个事务就可以和别的事务同时访问事务源了。四、分布式事务 一提到分布式事务，大家可能都想到有多个数据库分布在不同的计算机上。其实，不仅如此，例如一个事务需要同时控制对本地的数据库操作和本地的 JMS队列，也可以称为分布式事务。在EJB容器中，例如JBoss，我们会配置一些数据源，一些JMS队列等，无论采用BMP(Bean管理事务)还是 CMP(容器管理事务)，我们在真正使用的是JTA(Java Transaction API)的UserTransaction，它就是一个分布式事务，

它可以控制容器中所有的事务源。但是在Spring的应用中，要想使用分布式事务就有点困难了，好在还有JTOM，它可以做到分布式事务的控制，最终我们可以使用JOTM提供的UserTransaction，而原来的事务控制部分只需要很小的修改。我所经历的一个项目就是从JBoss迁移到Jetty中去，因为Jetty这样的Web容器不支持分布式事务，问题也就暴露出来了。所以如果大家在项目中需要迁移一个EJB项目到Spring中，千万别忽略了分布式事务的控制。