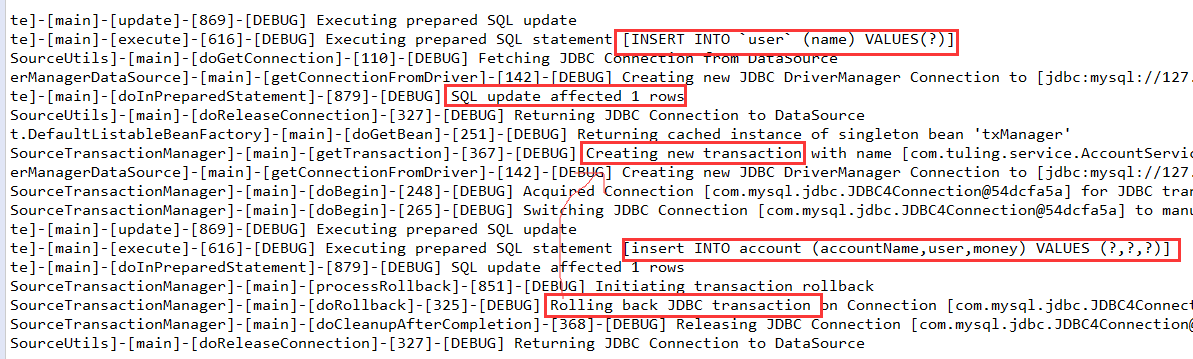
**关于事物的场景分析**

|  |  |  |  |
| --- | --- | --- | --- |
|  | createUser | addAccount(异常) | 预测结果 |
| 场景一 | 无事物 | required | user （成功） Account（不成功） |
| 场景二 | required | 无事物 | user （不成功） Account（不成功） |
| 场景三 | required | not\_supported | user （不成功） Account（成功） |
| 场景四 | required | required\_new | user （不成功） Account（不成功） |
| 场景五 | required  (异常移至createUser方法未尾) | required\_new | user （不成功） Account（成功） |
| 场景六 | required  (异常移至createUser方法未尾)  （addAccount 方法称至当前类）this.addAccount | required\_new | user （不成功） Account（不成功） |
| 场景七 | required  (异常移至createUser方法未尾)  （addAccount 方法称至当前类），**userService**.addAccount | required\_new | user （不成功） Account（成功） |

场景一

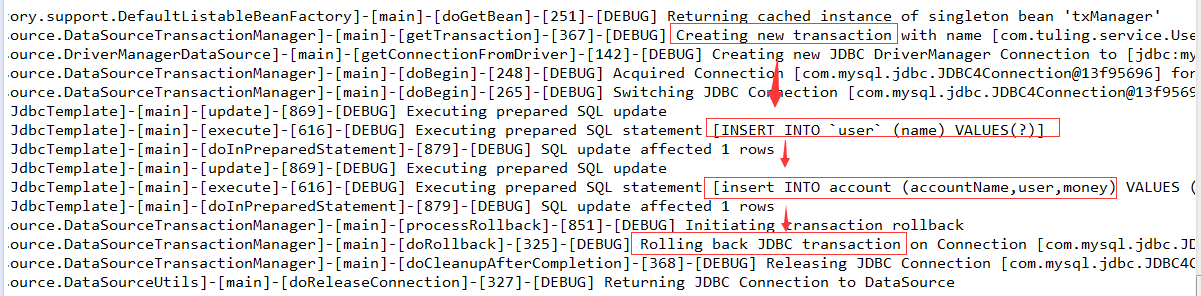
createUser成功，addAccount回滚



总结：外部方法无事物，内部方法回滚不影响外部方法

场景二

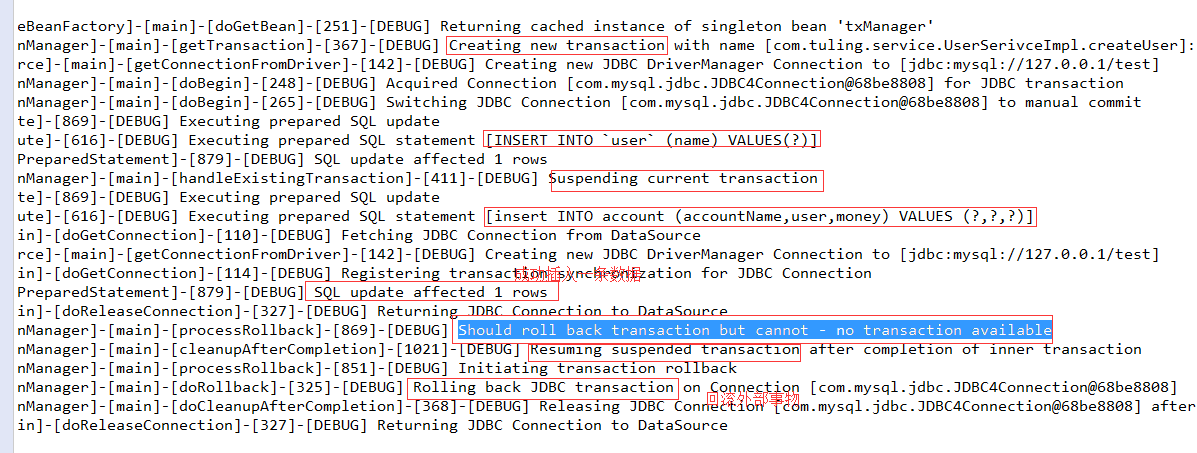
createUser和 addAccount在同一个事物，addAccount抛异常，回滚整个事物



场景三

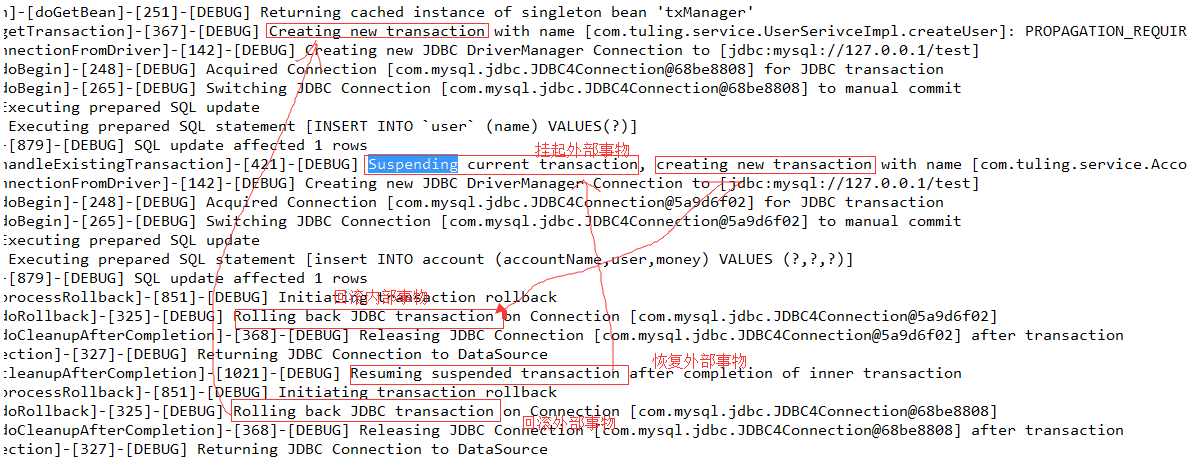
createUser回滚，addAccount不回滚

外部方法开启事物 –> 执行外部方法业务逻辑 –> 挂起外部事物 –> 执行内部方法业务逻辑，抛异常 –> 内部没有事物，不需要回滚 -> 唤醒外部事物 –> 回滚外部事物



场景四

createUser回滚，addAccount回滚

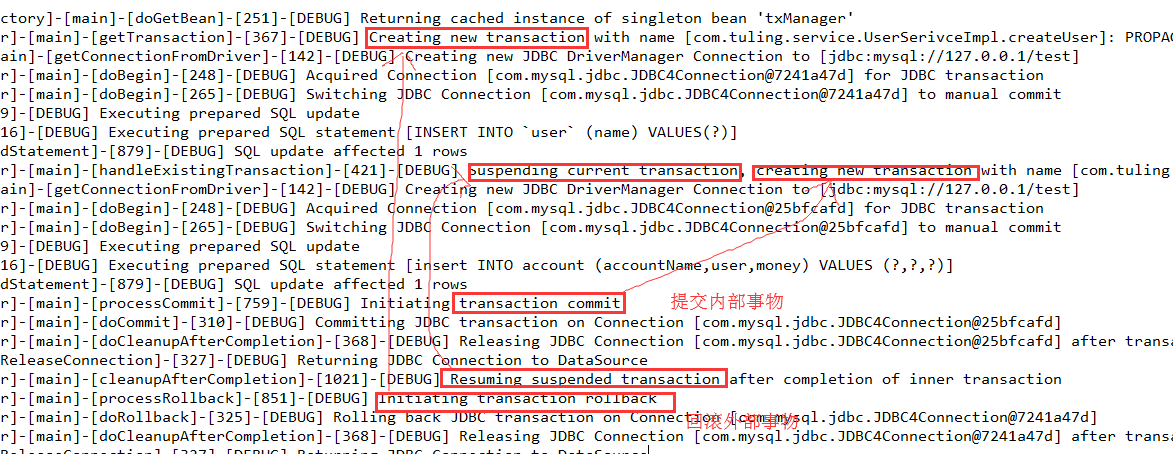


外部事物回滚的原因，可以自己断点调试下



场景五

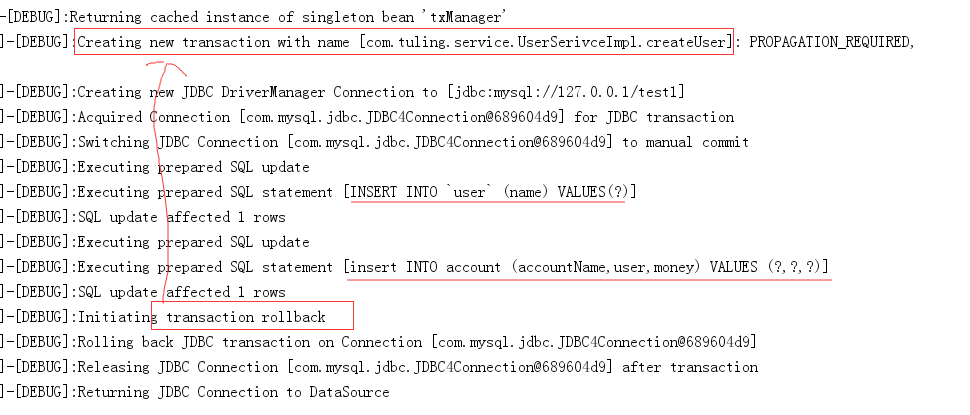
createUser回滚，addAccount提交成功



场景六

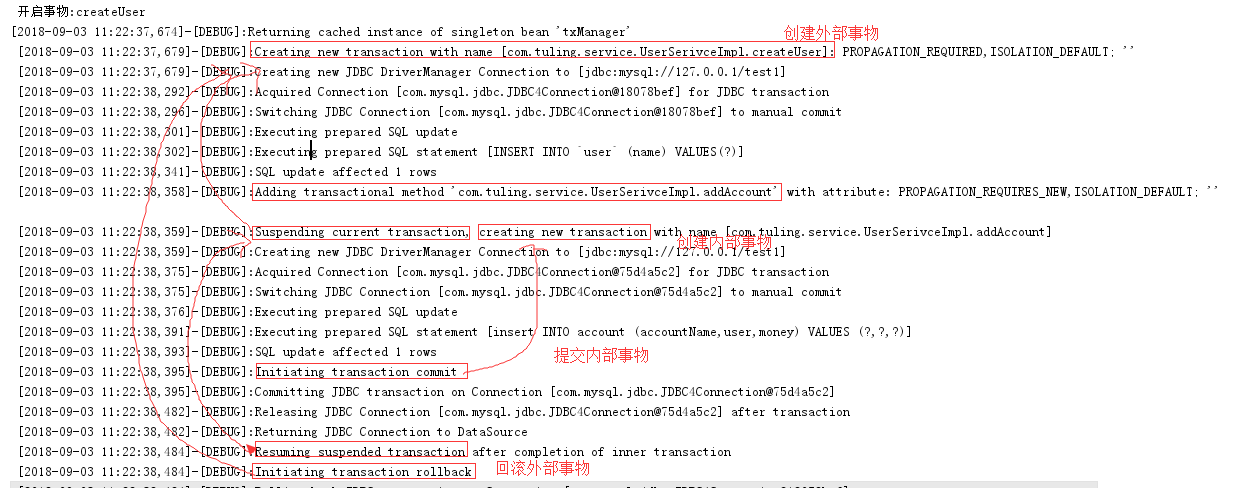
createUser开启事物，回滚事物

addAccount 没有被代理到



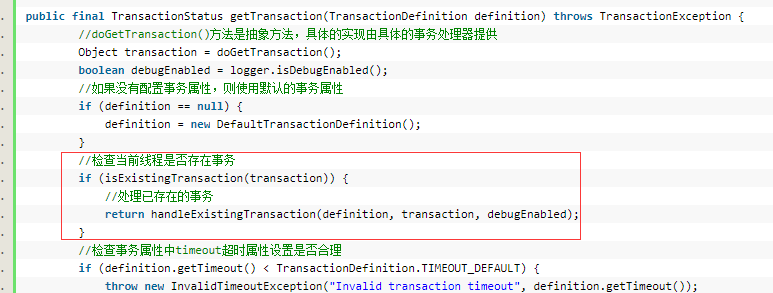
场景七

createUser回滚，addAccount提交成功



源码：

获取事物



不同事物的回滚逻辑



参考资料

[**《Spring技术内幕》学习笔记15——Spring事务处理的实现**](https://blog.csdn.net/chjttony/article/details/6528344)