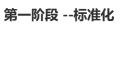


问题2:深入理解Spring的加载机制. --BeanDefinition 的创建 和bean实例的创建 --销毁的过程(bean的生命周期)





事务是基于AOP机制诞生的。一旦希望某个执行出现异常回滚,那么就一定需要准备好一个同一个数据库连接 // --- 不用框架提供的 事务管理功能 自己 JAVA应用程序 // @Transactional public void deleteUser(String id) throws Exception { Connection = dataSource.getConnection(); connection.setAutoCommit(false);// 改为手动提交 Statement statement = connection.createStatement(); connection try (// 插入用户表 -- userMapper.delete 数据库连接 数据库的回滚需要 知道这个连接提供 了多少SQL statement.execute(sql: "delete from tb_user where user_id ='" + id + "'"); // 插入日志记录表 -- loggerMapper.insert statement.execute(sql: "insert tb_log values('" + UUID.randomUUID() + "','删除用户: " + id + "', now())"); 数据源我们需要直接去拿就可以了 提供了无数的数据源实现 // 人造异常 数据库 int i = 1 / 0; System. out. println("提交"); connection.commit(); catch (Exception e) { 原生的方式回滚只需要 e.printStackTrace(); System. out. println("回滚"); rServiceImpl > deleteUser() 保存一个链接 思考,如果是orm框架,这个Spring事务会干点什么事情呢?如何去保证它的mapper也是一个连接? 最初的类 绑定参数生成 SQL 需要进行事务处理的具体方法 方法调用 数据库 获取数据库连接,执行SQL method pojo对象映 射结果