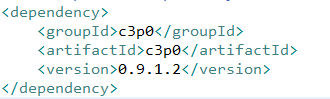
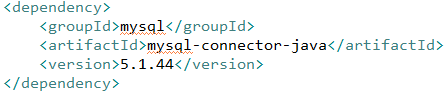
**CAP11 声明式事务**

1. 环境搭建：导入数据源、数据库驱动、Spring-jdbc依赖

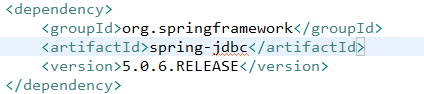
在pom.xml新增c3p0依赖包(c3p0封装了jdbc 对DataSource接口的实现)



在pom.xml新增数据库驱动依赖包



在pom.xml新增spring-jdbc依赖包（jdbcTemplate）



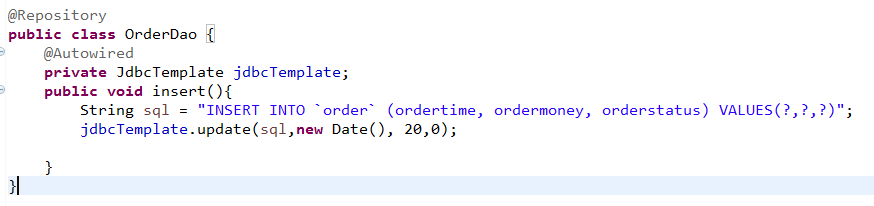
1. 新建Cap10MainConfig,将数据源和操作模板加载到IOC容器中



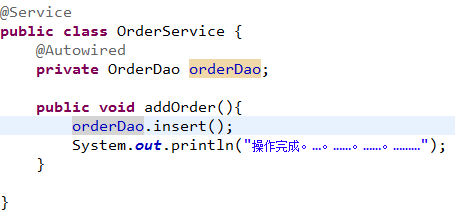
3，新建Order测试表

CREATE TABLE `order` (  
  `orderid` int(11) DEFAULT NULL,  
  `ordertime` datetime DEFAULT NULL,  
  `ordermoney` decimal(20,0) DEFAULT NULL,  
  `orderstatus` char(1) DEFAULT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=utf8

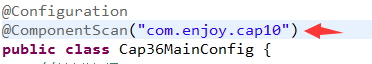
4，新建OrderDao操作数据库类



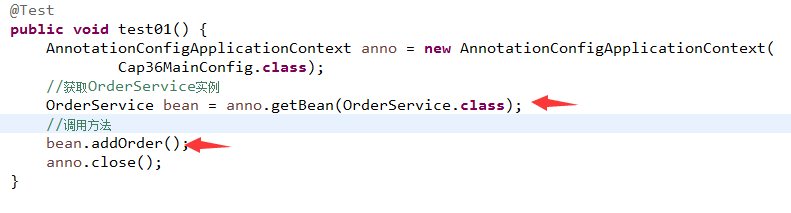
5，新建 OrderService类，将orderDao注入进来



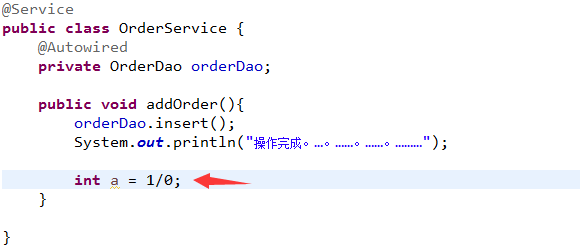
6，将OrderService和OrderDao扫描进来，加载到容器



7，新建测试用例进入测试（**测试结果：正常向数据库插入了一条记录**）

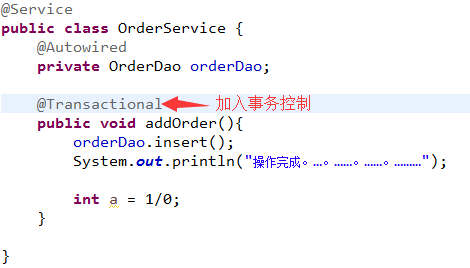


8，对OrderService引入异常，看是否还能插入数据库？



测试结果：报异常， 但数据库正常插入。

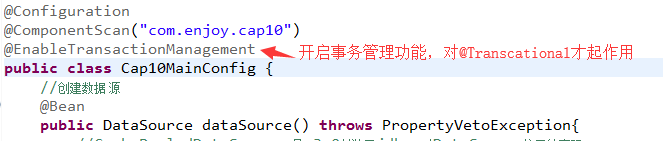
9，可以给OrderService添加事务，若出现异常，看是否能全部回滚？



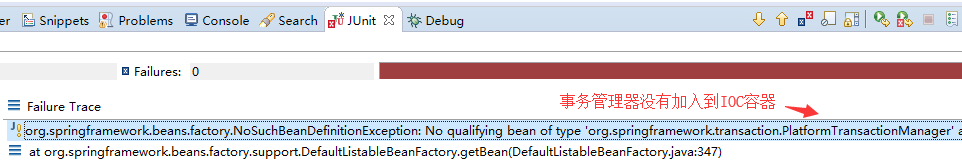
测试结果：事务不起作用， 照样可以成功插入

1. 基于上以分析，其实我们之前在xml配置里， 会配置开启基于注解的事务管理功能

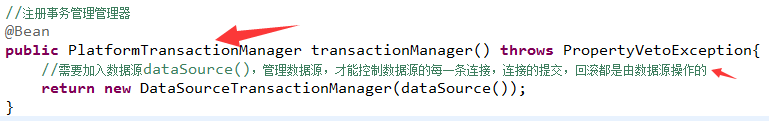
@EnableTransactionalManagement开启基于注解的事务管理功能，和AOP一样Enablexx



再测试，测试结果为：出错啦…………IOC容器没有这个事务管理器bean。



11，将事务管理器的bean加载到容器中，修改配置类，新增以下注册



再测试，测试结果：**正常回滚，没有报错**

12, @EnableTransactionManagement源码分析（与AOP的创建拦截流程一致）：

1）、@EnableTransactionManagement

利用TransactionManagementConfigurationSelector给容器中会导入组件

导入两个组件

AutoProxyRegistrar

ProxyTransactionManagementConfiguration

2）、AutoProxyRegistrar：

给容器中注册一个 InfrastructureAdvisorAutoProxyCreator 组件；基本的动态代理创建器

InfrastructureAdvisorAutoProxyCreator：？

利用后置处理器机制在对象创建以后，包装对象，返回一个代理对象（增强器），代理对象执行方法利用拦截器链进行调用；

3）、ProxyTransactionManagementConfiguration 做了什么？

1、给容器中注册事务增强器；

1）、事务增强器要用事务注解的信息，AnnotationTransactionAttributeSource解析事务注解

2）、事务拦截器：

TransactionInterceptor；保存了事务属性信息，事务管理器；

他是一个 MethodInterceptor；

在目标方法执行的时候；

执行拦截器链；

事务拦截器：

1）、先获取事务相关的属性

2）、再获取PlatformTransactionManager，如果事先没有添加指定任何transactionmanger

最终会从容器中按照类型获取一个PlatformTransactionManager；

3）、执行目标方法

如果异常，获取到事务管理器，利用事务管理回滚操作；

如果正常，利用事务管理器，提交事务