**Spring基础jar包：**

**1. spring-beans**

**2. spring-context**

**3. spring-core**

**4. spring-expression**

**5. commons-logging-1.2.jar**

**6. log4j-1.2.17.jar**

**Spring JDBC模板开发包：**

**1. spring-jdbc**

**2. spring-tx**

**MySQL数据库驱动jar：**

**1. mysql-connector-java-5.1.46.jar**

**c3p0相关的jar包：**

**1. c3p0-0.9.2.1.jar**

**2. mchange-commons-java-0.2.3.4.jar**

**Spring AOP开发包：**

**1. aopalliance-1.0.jar**

**2. aspectjweaver-1.8.7.jar**

**3. spring-aop**

**4. spring-aspects**

**junit相关的jar包：**

**1. junit-4.12.jar**

**2. hamcrest-core-1.3.jar**

**1，**spring是开源的轻量级框架，模块化非侵入式的设计；

**2，**spring核心主要两部分：

2004年Martin Fowler在他的网站上提出了关于控制反转（IoC，Inversion of Control）的问题，“The question is, what aspect of control are [they] inverting?”，后来，他又建议重新命名这项原则，使其可以自我解释，从而提出了依赖注入（DI，Dependency Injection）的概念。

**Spring以IoC、AOP为主要思想，其中IoC，Inversion of Control 指控制反转或反向控制。在Spring框架中我们通过配置创建类对象，由Spring在运行阶段实例化、组装对象。AOP，Aspect Oriented Programming，面向切面编程，其思想是在执行某些代码前执行另外的代码，使程序更灵活、扩展性更好，可以随便地添加、删除某些功能。Servlet中的Filter便是一种AOP思想的实现。**

a，aop（Aspect Oriented Programming）：面向切面编程，拓展功能不是修改源代码实现；

b，ioc（Inversion of Control）：控制反转；

比如有一个类，在类里面有方法（不是静态的方法），调用类里面的方法，需要创建类的对象，适用对象调用方法，创建类对象的过程，需要new出来对象，而ioc不是通过new方式实现，而是交给spring配置创建类对象；

**3，**spring是一站式框架；

a，spring在Javaee三层框架中，每一层都提供不同的解决技术；

3.1 Web层：springMVC；表现层

3.2 Service层：spring的ioc；业务逻辑层

3.3 Dao层：spring的jdbc Template；数据访问层

**4**，spring版本

a，spring4.x

**spring中的BeanFactory和FactoryBean的区别与联系：**

首先，这俩都是个接口；实现 BeanFactory 接口的类表明此类是一个工厂，作用就是配置、新建、管理各种Bean；而实现 FactoryBean 的类表明此类也是一个Bean，类型为工厂Bean（Spring中共有两种bean，一种为普通bean，另一种则为工厂bean）。顾名思义，它也是用来管理Bean的，而它本身由spring管理；其实Bean主要是通过自己定义一个BeanFactory工厂的实现，通过构造函数传入初始值，读取XML配置文件信息。并且将读取到的id作为key值，读取到的对象作为Value值，存入到Map容器中。当客户端需要获取时，只要传入id，就能从容器中读取出来；

**什么是bean？**

规律：凡是子类及带有方法或属性的类都要加上注册Bean到Spring IoC的注解；把Bean理解为类的代理或代言人（实际上确实是通过反射、代理来实现的），这样它就能代表类拥有该拥有的东西了；@Bean就放在方法上，就是产生一个Bean；

<https://www.cnblogs.com/bossen/p/5824067.html>

**spring的ioc操作：**

1，把对象的创建交给spring进行管理；

2，ioc操作两部分：

a，ioc文件配置文件方式；

b，ioc注解方式；

**ioc底层原理：**

1，ioc底层原理使用技术

a，xml配置文件；

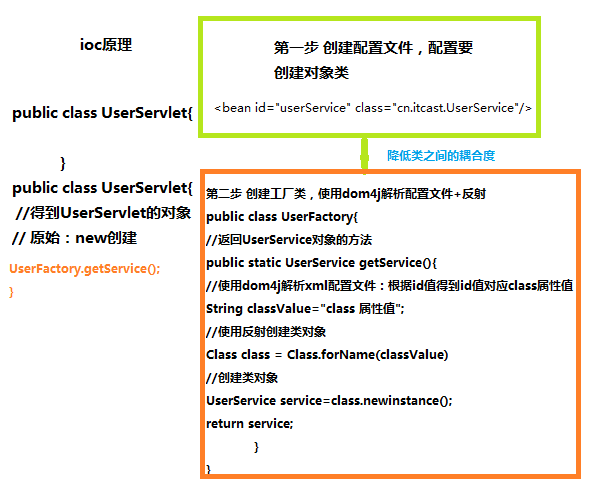
b，dom4j解决xml；

c，工厂设计模式；

d，反射；

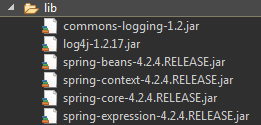
**画图分析ioc的实现原理：**



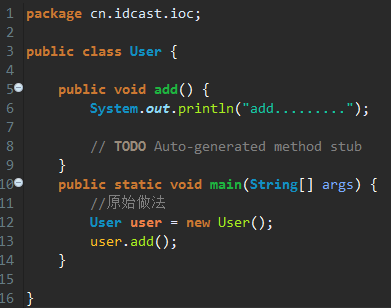


**Ioc入门案例：**

第一步 导入jar包



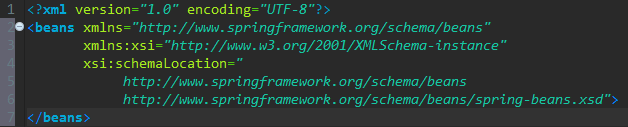
第二步 创建一个类，在类里面创建方法



第三步 创建spring配置文件，配置创建类

a，spring核心配置文件名称和位置是不固定的：建议放在scr下，官方建议applicationContext.xml

b，引入schema约束



c，配置对象创建：



第四步 写代码测试对象创建

a，代码只在测试中使用；

**配置文件没有提示问题**

a，spring引入schema约束，把约束文件引入到Eclipse中；

1. [*http://www.springframework.org/schema/beans/spring-beans.xsd*](http://www.springframework.org/schema/beans/spring-beans.xsd)

**spring中的bean管理（xml方式）：**

Bean实例化的方式：

**a，**在spring里面通过配置文件创建对象；

**b，**bean的实例化的三种方式；

a）使用类的无参数构造创建；类里面没有无参构造函数会出现异常；如：

<!-- 等同于 user = new com.wm103.ioc.User(); -->

<bean id="user" class="com.wm103.ioc.User"></bean>

b）使用静态工厂创建；1）创建静态方法，返回类对象



c）使用实例工厂创建；1）创建一个不是静态的方法，返回类对象；

**Bean标签常用属性：**

1，id属性：起名称，理论上可以任意命名，id属性值不应为中文及特殊符号，可以根据id值得到配置属性；

2，class属性：创建对象所在类的全路径；

3，name属性：功能和id属性一致，但name属性值可以包含一些特殊符号；

4，scope属性：singleton：默认值，单例对象

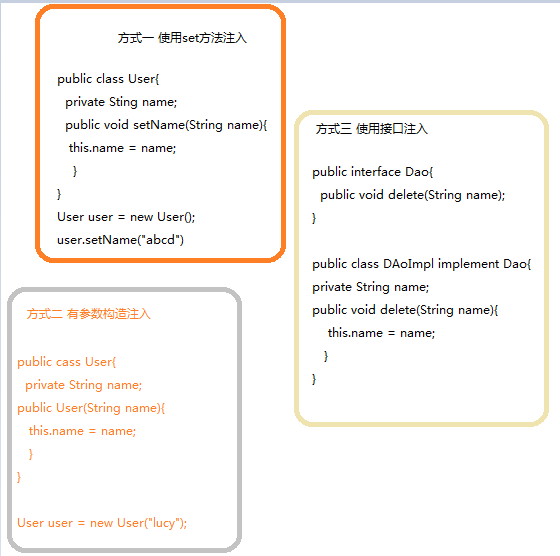
prototype：多实例对象

**属性注入：**

1，传感对象时，向类的属性里注入值，即给类的属性设置值；

2，属性注入的方式：在spring框架中仅支持前两种属性注入方式；

在Spring框架中支持set方法注入和有参构造函数注入，即创建对象后通过set方法设置属性或采用有参构造函数创建对象并初始化属性。



1，使用有参数构造注入属性：

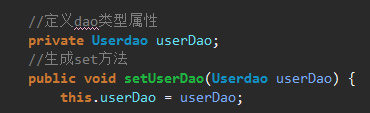
2，使用set方法注入属性；

**注入对象类型属性（重点）：**

1，创建Service类和dao类：

具体实现过程：1），在service里面把dao作为类型属性；

2）生成dao类型属性的set方法；



3）在配置文件中完成注入关系：

**注释：如果@Resource出错，可以尝试进入project>properties>project facets>修改java版本为1.6**



**IOC与DI区别：**

1）IOC控制反转，将传统的对象创建流程转变为交由框架进行创建和管理。在Spring中，对象的创建交给Spring进行配置。它包括依赖注入；

2） DI：依赖注入，向类里面的属性中设置值；

关系：DI不能单独存在，需要在IOC的基础之上完成操作；

**Spring整合web项目管理：**

1）new对象，功能可以实现，但效率较低；



1，实现思想，把家在配置文件和创建对象过程，在服务器启动时完成；

2，实现原理，

（1）servletcontext对象

（2）监听器

具体使用：

在服务器启动时候，为每个项目创建servletcontext对象（仅有一个）；

在servletcontext对象创建时，使用监听器可以知道SC对象在什么时候创建；

使用监听器监听到SC对象创建时，监听器会加载spring配置文件，把配置文件中配置对象进行创建；

把创建出的对象放到SC域对象里面（用setattribute方法）；

获取到对象时候，到SC域里面得到（用getattribute方法）；

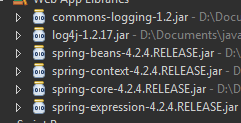
**Spring的bean管理（注解）：**

**注解：代码里的特殊标记，使用注解可以完成相关功能；写法为@注解名称（属性名称-属性值）；可以用在方法、类、属性上面；**

Spring注解开发准备工作：

1，导入jar包；

1）导入基本的jar包；



2）导入aop的jar包；



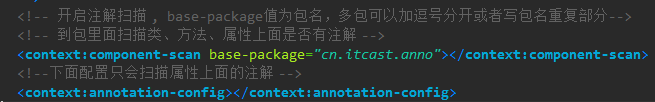
2，创建一个类，创建一个方法；

3，创建spring的配置文件，引入约束

1）前面的ioc功能的约束是beans约束，做spring ioc的注解开发，还需要引入新的约束；



4，开启注解扫描：只需要下面两个配置的一个就可；



注解创建对象：

1,）在要创建对象的类上面使用注解实现；用value值测试；



1）@component：

2）@controlller： web层

3）@service： 业务层

4）@repository：持久层

四个注解的功能是一样的，都创建对象；后三个注解是为了后期增强；

创建对象是单实例还是多实例：



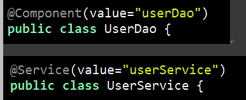
**注解注入属性：**

1，创建service类，创建dao类，在service里得到dao对象：

注入属性第一个注解是@autowired（此注解查找类名），第二个是@resource：



1）创建dao和service类对象：



2）在service类里面定义dao属性：



**配置文件和注解混合使用：**

**1，创建对象操作使用配置文件方式实现：**

**2，注入属性操作使用注解方式实现：**

**aop：**

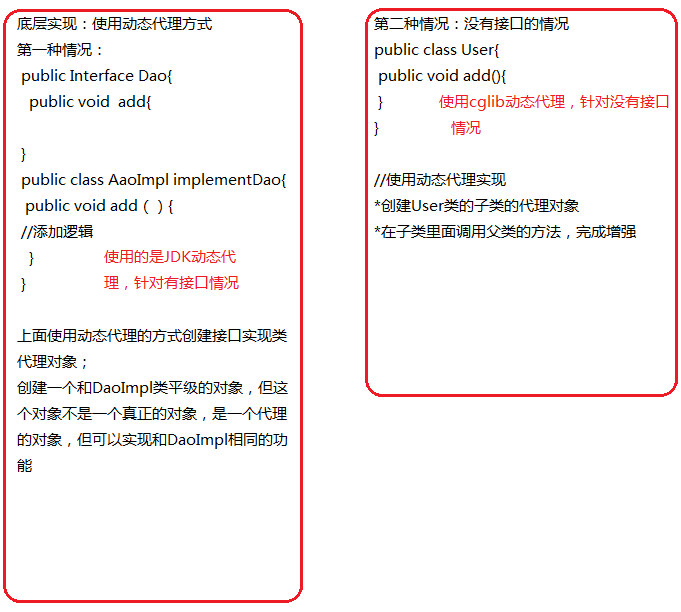
AOP，即面向切面编程，简单来说其思想指的是在不修改原有代码的情况下扩展新的功能。在传统OOP思想下，我们扩展一个类的功能，可能采取的方式为纵向继承，即定义父类，编写新功能，通过继承的形式在子类中使用父类的方法扩展功能。这种方式下，存在着类与类之间的耦合，多个类需要扩展多个新的功能，都需要新增新的代码到原有类中。

　　而AOP思想则表示采用“在程序运行期间将某段代码，动态的切入到某个类的指定方法的指定位置”的方式，即我们不需要对原有类的功能进行修改，而是在程序运行过程中，将新增的功能加入都被扩展的类型的指定方法的指定位置**。**

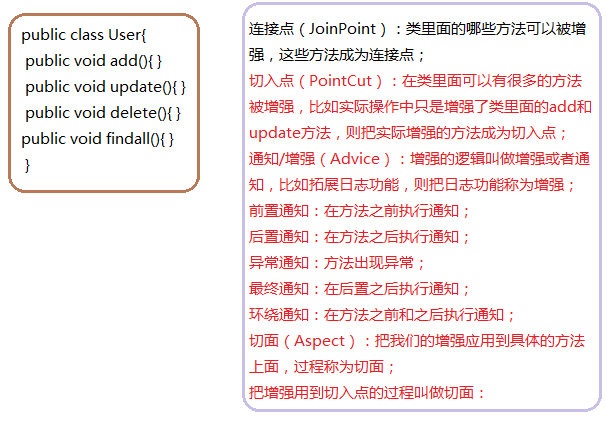
**aop概念：**面向切面编程，拓展功能不通过修改源代码实现；

aop采取横向抽取机制，取代了传统纵向继承体制重复性代码；

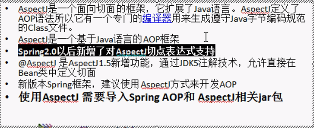
**aop原理：**



**Aop操作中的相关术语：**



**1，在spring里面进行aop操作，使用aspectj实现：spring2.0之后才增加了对aspectj的支持；**



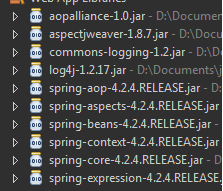
**2，使用aspectj方式实现aop操作有两种方法：**

1）基于aspectj的xml配置：

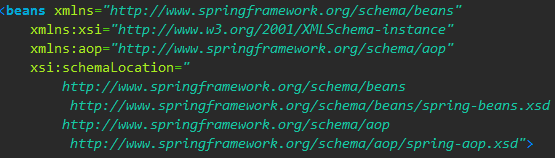
2）基于aspect的注解方式：

**Aop操作的准备：**

1）除了需要导入基本的jar包之外，还需要导入aop相关的jar包；



2）创建spring的核心配置文件，导入aop的约束；



**使用表达式配置切入点：**

1）切入点，实际增强的方法；

2）常用的表达式：(\*表示任意修饰符)

全路名：包名，类名，方法名），如果加\*代表具体的方法名，表示对类中所有方法增强，（..）表示参数；

Execution（<访问修饰符>?<返回类型><方法名>（<参数>）<异常>）

举例几个主要使用：

1. execution(public \* \*(..))：匹配所有类的public方法；

2. execution(\* com.wm103.dao.\*(..))：匹配指定包（这里指com.wm103.dao）下的所有类的方法，不包含子包；

3. execution(\* com.wm103.dao..\*(..))：匹配指定包以及其下子孙包下的所有类的方法，值得注意的是，“..”出现在类名中时，后面必须跟“\*”，表示包、子孙包下的所有类；

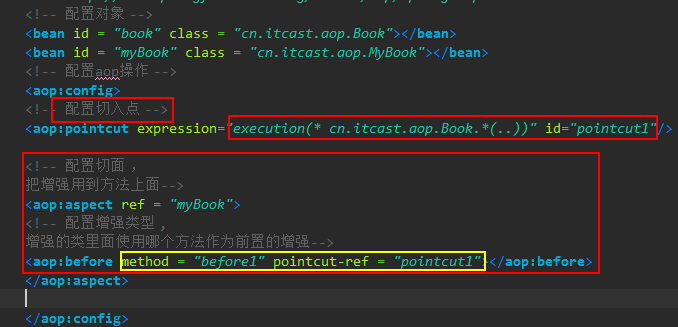
4. execution(\* com.wm103.dao.UserDaoImpl.\*(..))：匹配指定类下的所有方法；

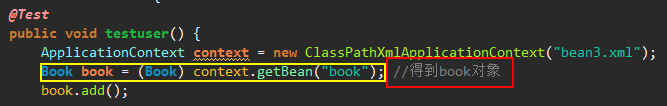
5. execution(\* com.wm103.dao.UserDao+.\*(..))：匹配实现指定接口（这里指com.wm103.dao.UserDao接口）的所有类的方法；

6. execution(\* save\*(..))：匹配所有指定前缀（这里指的是save）开头的方法；

7. execution(\* \*.\*(..))：匹配所有类的所有方法;

**Aspectj的aop操作：**







增强的顺序默认按照xml中配置的顺序运行。如果需要将TimeHandler增强先执行的话，那么可以调整LogHandler和TimeHandler的顺序。或者在aop:aspect标签中使用order属性（按从小到大的顺序执行）。



**Log4j介绍：**



1，通过log4就可以看到程序运行过程中更详细的信息；

1）经常使用log4j查看日志；

2，log4j的使用：

1）首先导入log4j的jar包；



2）复制log4j的配置文件，复制到src下面；



3）设置日志级别：



a）看到基本信息；

b）debug，看到更详细信息；

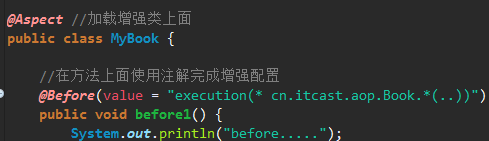
**基于aspectj的方式注解aop操作：**

1，使用注解方式创建aop对象；

2，在spring的核心配置文件中，开启aop注解扫描的操作；



3，在增强的类上面实现注解；



**Spring的jdbctemplate操作：**

1，spring框架是一站式框架，对阵javaee三层中，每一层都要解决技术。在dao层，使用JdbcTemplate；

2，JdbcTemplate对jdbc进行了封装；

3，JdbcTemplate使用和dbutils使用很相似，都对数据库进行crud操作；

**使用步骤：**

增加：

1，导入jdbctemplate使用的jar包：



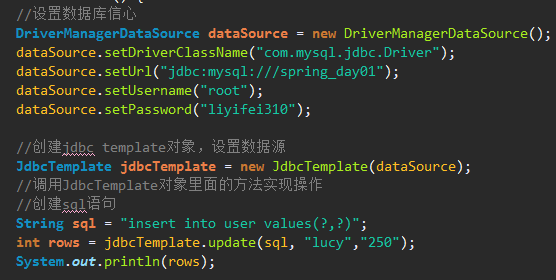
添加mysql驱动数据包：



2，设置数据库信息，

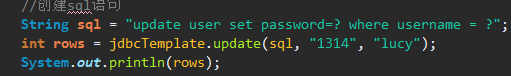
3，创建JdbcTemplte对象，设置数据源；

4，调用JdbcTemplate对象里面的方法实现操作；



通过以上简单案例，我们使用Spring中的JdbcTemplate（即对JDBC提供的封装）实现了添加操作。但在该案例中实际上存在着一个问题，即DriverManagerDataSource对象在UserDaoImpl类中被创建，无法为Dao层中其他的操作数据库的实现使用，除此之外，DriverManagerDataSource底层获取数据库连接是通过DriverManager.getConnection获取，每次调用DriverManagerDataSource的getConnection获取对数据库的连接，都相当于创建一个新的连接，这种方式下耗费内存和时间，实用性低。我们需要一个数据库连接池，能有效地负责创建、管理和分配数据库连接。为了方便对Spring的JdbcTemplate进行讲解，仍采用这种形式创建数据源。后续将介绍c3p0连接池的使用（c3p0实现了DataSource接口，维护了数据库连接，负责创建、管理和分配数据库连接）。

修改：



删除：



查询：

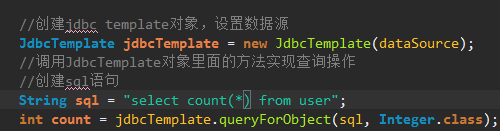


查询的具体实现：

1）查询返回某一个值：



第一个参数是sql语句，第二个参数是返回类型的class；



2）查询返回某一个对象：





3）查询返回list集合：

，

参数一为sql语句，第二个是Rowmapper接口，自己写类实现数据封装；，第三个是可变参数；



**Spring配置连接池和dao使用jdbctemplate：**

1，spring配置c3p0连接池：

第一步，导入jar包；



第二步，配置spring文件，配置连接池；代码中实现如下：



把代码中实现，在配置文件中配置出来：

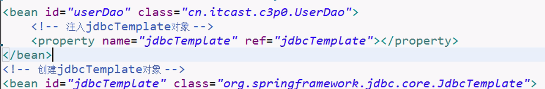


配置文件对应的是上面的代码；

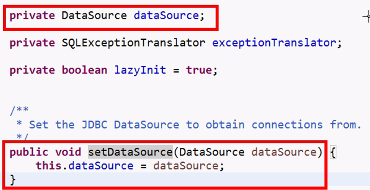
**Dao中使用JdbcTemplate模板：**

1，创建service和dao，配置了service和dao对象，在service里面注入了dao对象：

2，创建一个JdbcTemplate模板对象，把模板对象注入到dao中：



3，在我们JdbcTemplate对象里面注入datasource：





**Spring的事务管理：**

**事务概念：**

1，什么是事务？

在关系数据库中，一个事务可以是一条SQL语句，一组SQL语句或整个程序；事务由事务开始(begin transaction)和事务结束(end transaction)之间执行的全体操作组成；

2，事务的特性：原子性，一致性，隔离性，持久性；

原子性（atomicity）：一个事务是一个不可分割的工作单位，事务中包括的诸操作要么都做，要么都不做；

一致性（consistency）：事务必须是使数据库从一个一致性状态变到另一个一致性状态。一致性与原子性是密切相关的；

隔离性（isolation）：一个事务的执行不能被其他事务干扰。即一个事务内部的操作及使用的数据对并发的其他事务是隔离的，并发执行的各个事务之间不能互相干扰；

持久性（durability）：持久性也称永久性（permanence），指一个事务一旦提交，它对数据库中数据的改变就应该是永久性的。接下来的其他操作或故障不应该对其有任何影响；

3，不考虑隔离性会产生读问题；

4，解决读问题：设置隔离级别；

**Spring事务管理api：**

1，事务管理两种方式：

1）编程式事务管理：

2）声明式事务管理：

a) 基于xml配置文件方式；b) 基于注解方式；

2，spring进行事务管理中的api介绍：

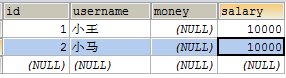


1）spring针对不同的dao层框架，提供了不同的实现类；

2）首先配置事务管理器；

**搭建转账环境：**

第一步：创建数据库，添加数据进去；



第二步：创建一个service和一个dao的类，完成注入的关系；

1）service层又叫业务逻辑层；

2）dao层是单纯对数据库操作层，在dao层不添加业务；

3）小王转账1000块给小马；



3，产生问题：

C:\Users\ADMINI~1\AppData\Local\Temp\1522054462.png

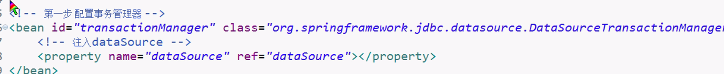
4，解决

1）添加事务解决，出现异常进行回滚操作；

**声明式事务管理（xml配置）：**

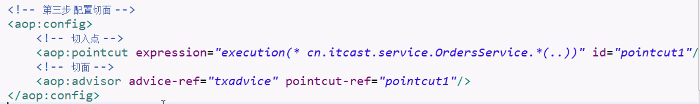
**1，配置文件方式使用aop思想配置”**

第一步 配置事务管理器



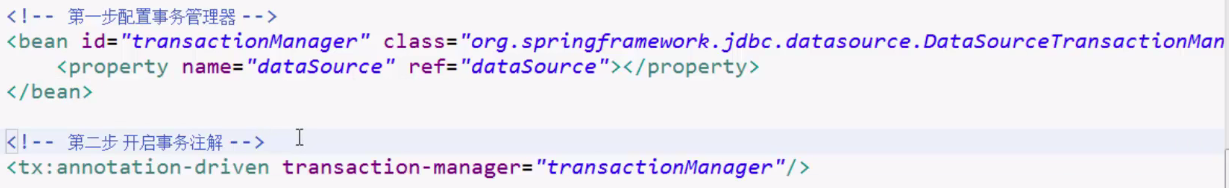
第二步 配置事务增强



第三部 配置切面：

**声明式事务管理（注解）：**

第一步 配置事务管理器



第二步 配置事务注解

第三部 在要使用事务的方法所在类上面添加一个注解



**SSH三大框架整合：**

Hibernate框架：

1，Hibernate核心配置文件：

1）orm思想：对象关系映射

