# Spring基本概念

1.轻量级开源框架

2.核心功能：

AOP，面向切面编程，扩展功能而不是修改源代码

IOC，控制反转

-比如一个类里有一个方法（非静态方法），调用类里的方法需要new此类创建一个对象，再调用

-对象创建不通过new方式，spring框架通过配置的方式实现

1. 一站式

spring在java的三层结构中都提供了不同的解决技术

-web层，springMVC

-service层，spring的ioc

-dao层，srping的JDBCTemplate

# Spring的IOC操作

1. 把对象的创建交给spring管理
2. Ioc操作两部分（创建对象的两种方式）
3. ioc的配置文件方式
4. Ioc的注释方式

## ioc底层原理

1. IOC底层使用的技术
2. xml配置文件
3. Dom4j解决xml
4. 工厂设计模式
5. 反射

## Spring的bean管理（xml方式）

Bean实例化的方式

1. 在Spring里通过配置文件创建对象
2. bean实例化的三种方式
3. 使用类的无参数构造创建

类里面要有无参数构造方法（默认只有无参数构造方法，若添加了有参数构造方法，就没有无参数构造方法了）

1. 使用静态工厂创建

创建静态的方法，返回类对象

1. 使用实例工厂创建

## bean标签常用属性

1. id属性

起名称，不能包含特殊符号，只能是纯英文，通过ID获取对象

1. class属性

创建对象所在类的全路径

1. name属性

和id的功能一样，id可以用name代替，但用name的名称可以包含特殊符号

1. Scope属性

Singleton:默认值，单例

Prototype:多例

Request:

Session:

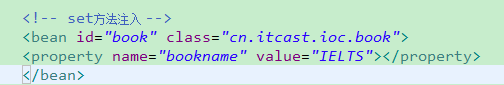
Globalsession:单点登录

## 属性注入：

创建对象时，给对象的属性赋值

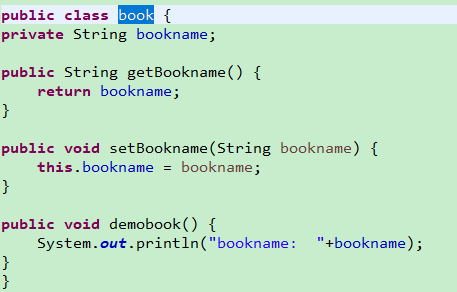
注入方式：

### （1）set方法注入 最常用



name属性值：类里面定义的属性名称

value属性值：要设置的值



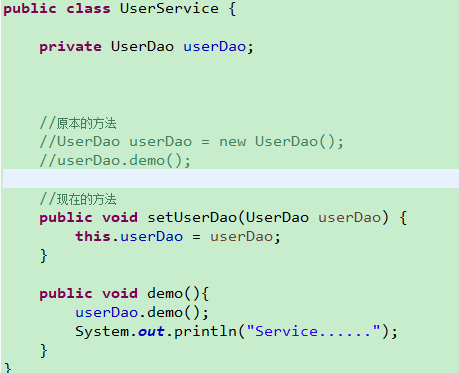
### （2）有参数构造方法注入

### （3）接口注入

在Spring框架中只支持前两种方法注入

注入对象类型属性（重点）





### p名称空间注入





## 注入复杂类型属性

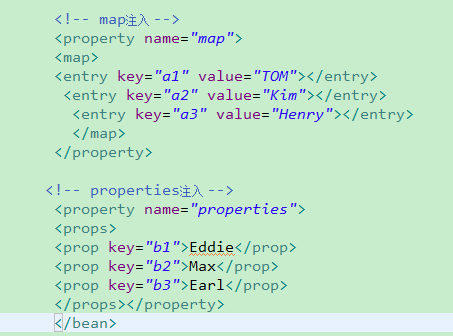
### （1）数组类型

### （2）List集合

### （3）Map集合

### （4）Properties类型





## IOC和DI区别

IOC：控制反转，把对象的创建交给spring

DI：依赖注入，向类里面的属性注入值

关系：DI不能单独存在，需要在IOC的基础上操作

# Spring整合Web项目

## 加载核心配置文件



New对象可以实现功能，但效率低下

## 实现思想

把加载配置文件和创建对象过程，在服务器启动时完成

## 实现原理

1. ServletContext对象
2. 监听器
3. 具体使用：

-在服务器启动时，为每个项目创建一个ServletContext对象

-在ServletContext对象创建时，使用监听器可以得到对象创建的具体时间

-使用监听器得到ServletContext对象创建的时间，

-加载Spring配置文件，把配置文件配置对象创建

-把创建的对象放到ServletContext对象的域里面（setAttribute方法）

-获取对象时，在ServletContext域里面得到（getAttribute方法）

# Spring的bean管理（注解方式）

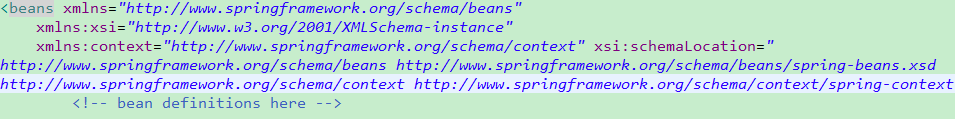
注解：1.代码里面的特殊标记，使用注解可以完成特殊功能

2.注解写法 @注解(注解属性名称=注解属性值)

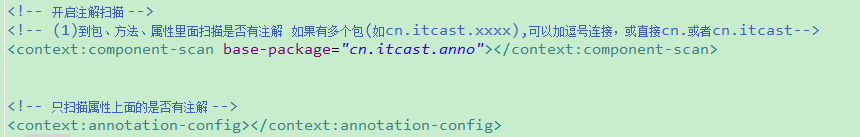
3.注解可以使用在类上面，属性上面，方法上面

## Spring注解开发准备

1. 导入jar包
2. 导入基本jar包（之前的6个）
3. 导入aop的jar包
4. 创建类，方法
5. 创建Spring配置文件，引入约束
6. 第一天做ioc基本功能时，引入bean约束
7. 做spring的注解开发，还要引入context约束

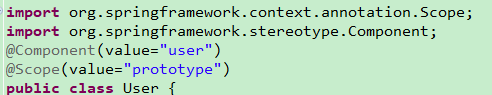


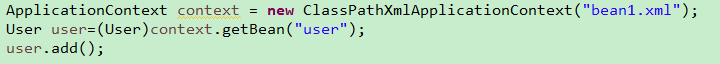
1. 开启注解扫描



### 使用注解创建对象

1. 在创建对象的类上使用注解实现





1. 创建对象有四个注解 目前来说功能一样

@Component

@Controller：Web层

@Service：业务层

@Repository：持久层

1. 创建的对象是单实例还是多实例



### 使用注解注入属性

1. 创建Service类、Dao类，在Service类中得到Dao类的对象

（1）创建Service类和Dao类对象





1. 在service类里定义Dao类属性

1.用@Autowired注解

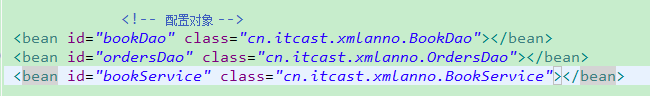


1. 用@Resource注解

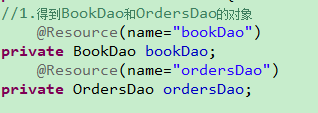


### xml和注解方式混合使用

1. 创建对象一般使用配置文件的方式



2.注入属性一般使用注解的方式



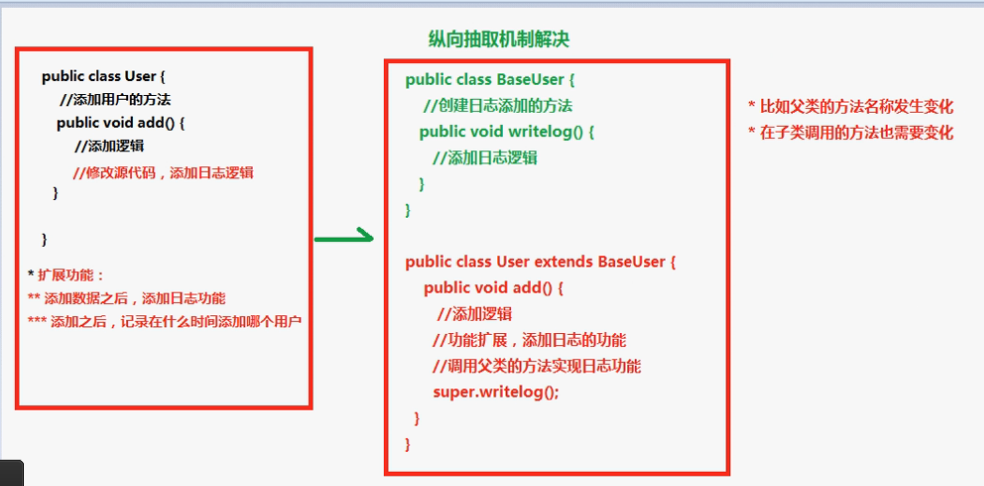
# AOP相关

## AOP概念

1. 面向切面（方面）编程，扩展功能不修改源代码
2. AOP采取横向抽取机制，取代了传统纵向继承体系重复性代码

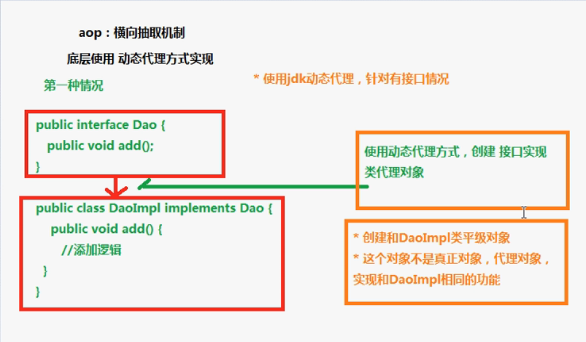
## AOP原理

### 旧的纵向抽取机制（继承父类方法）

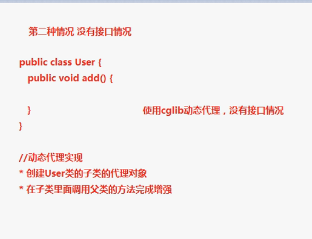


### 横向抽取

第一种情况：有接口

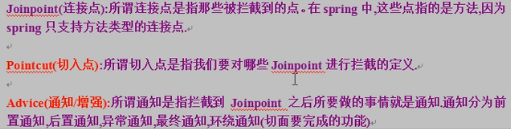


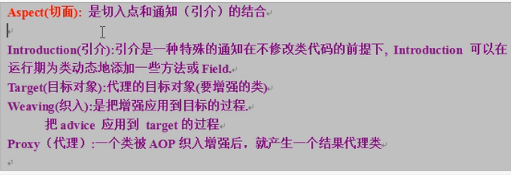
第二种情况：没有接口

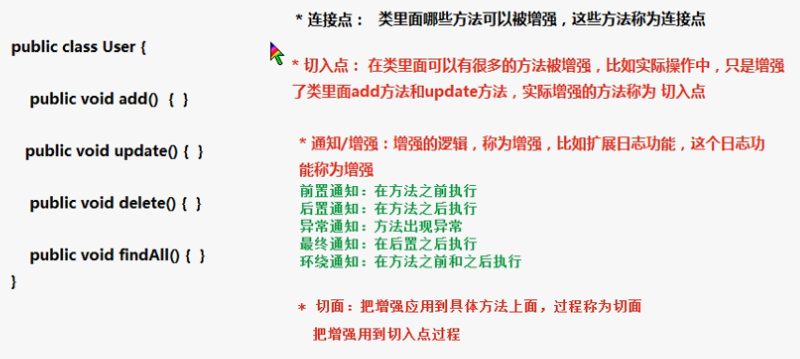
代理对象是user的子类

### AOP相关术语

重点掌握红色







# Spring的AOP操作

## 在Spring里进行AOP操作，使用AspectJ实现

（1）aspectj不是Spring的一部分，和Spring一起使用进行AOP操作

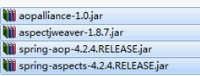
（2）Spring2.0以后新增了对Aspectj的支持

## 使用AspectJ实现AOP两种方式

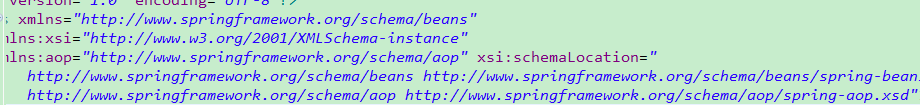
1. 基于aspectj的xml配置
2. 基于aspectj的注解方式

## AOP操作准备

1. 除了导入基本的jar包外，还要导入aop相关的jar包



1. 配置Spring文件，导入AOP约束



## 使用表达式配置切入点

### 切入点：实际增强的方法

### 常用的表达式



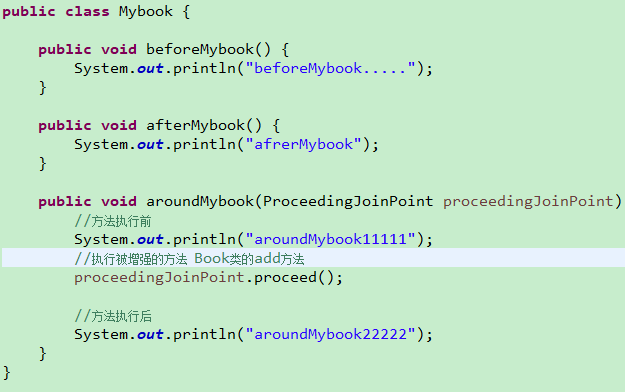
1. execution(\*cn.itcast.aop.Book.add(..)) \*表示所有类型(public..private..)

add(..) ..表示有无参数都可以

1. execution(\*cn.itcast.aop.Book.\*(..)) 类里面的所有方法
2. execution(\* \*.\*(..)) 所有类的所有方法
3. execution(\* save\*(..)) 匹配所有方法名是save开头的方法

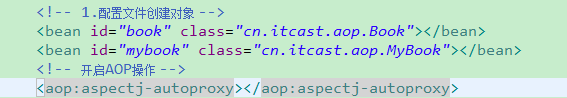
### Spring的AOP操作（基于XML配置文件）



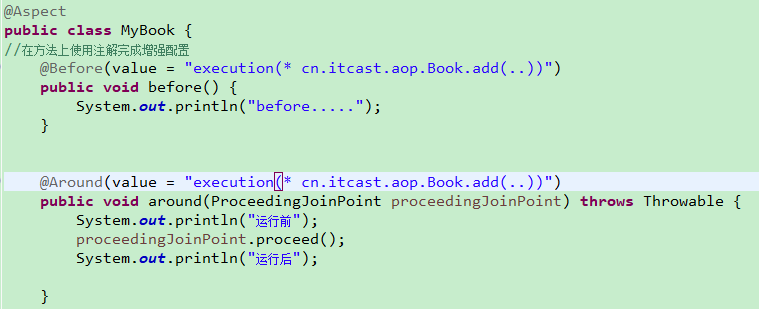


### Spring的AOP操作（基于Aspectj的注释）

1. 导入约束
2. 配置文件创建对象
3. 开启AOP操作



1. 用注解实现AOP操作



# log4j介绍

## 通过log4j可以看到程序运行过程更详细的信息

经常使用log4j查看日志

## log4j使用

1. 导入log4j的jar包
2. 复制log4j的配置文件，复制到src下面 log4j.properties

# Spring的jdbcTemplate操作

## Spring的一站式架构

1. 针对javaee三层，每一层都有解决技术
2. 在dao层，使用jbcTemplate

## 2.Spring对不同的持久化技术都进行了封装



### jdbcTemplate对jdbc进行封装

## jdbvTemplate使用和dbutils使用很相似，都对数据库进行crud操作

### （1）添加

1. 导入jdbcTemplate相关的jar包

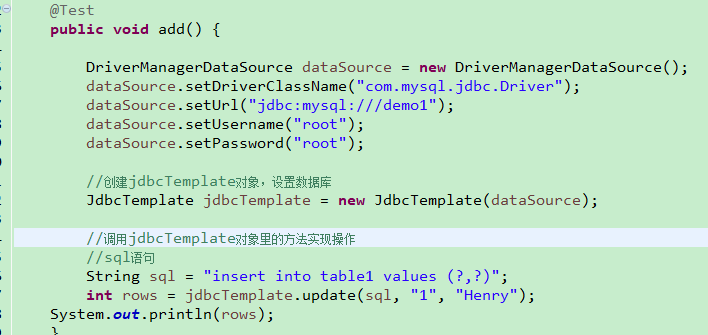


不要忘了数据库驱动的jar包！！！！

1. 设置数据库信息



1. 创建jdbcTemplate对象，设置数据库
2. 调用jdbcTemplate对象里的方法实现操作



（2）修改 同上

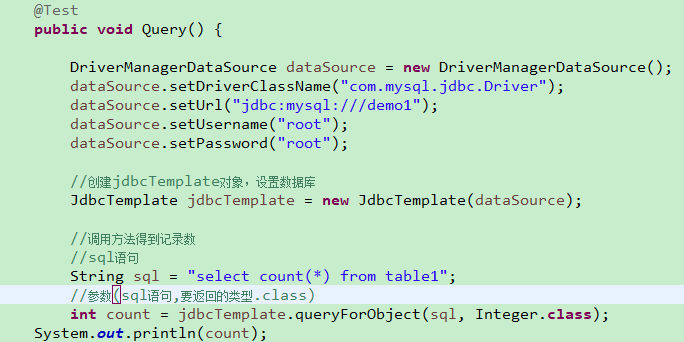
（3）删除 同上

### （4）查询



### 查询的具体实现

### 查询某一个值



### 查询返回某一个对象

先看JDBC原始方法



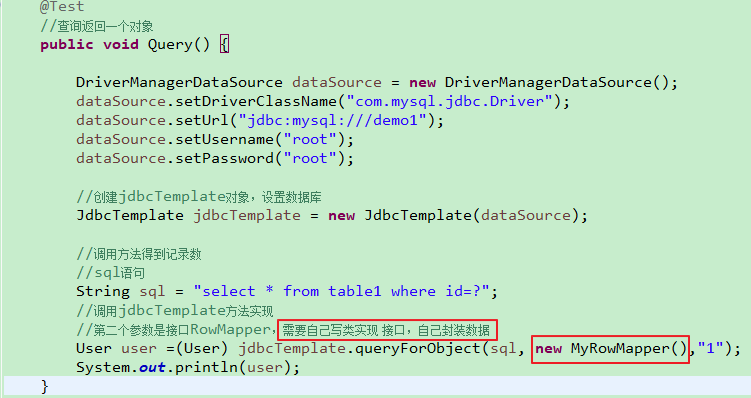
下面是jdbcTemplate方法



第一个参数是sql语句

第二个参数是RowMapper，是接口，类似于dbutils里的resultset

第三个参数是可变参数，对应sql语句的?



### 图像 56

### 查询返回一个List



1. sql语句
2. rowmapper接口，自己写类实现数据封装
3. 可变参数



# Spring配置连接池和dao使用jdbcTemplate

（service调用dao进行数据库操作）

## spring配置c3p0连接池

1. spring基本jar包+导入jar包



1. 创建Spring配置文件，配置连接池



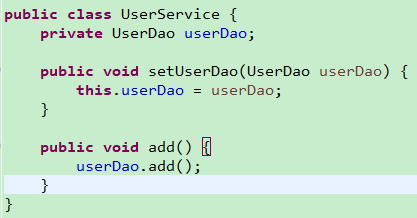
（3）把代码在配置文件中进行配置



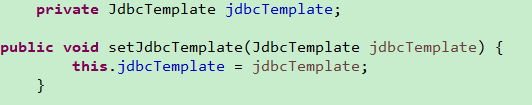
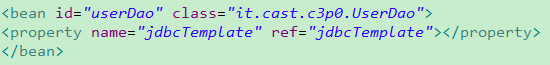
## dao使用jdbcTemplate

1. 创建Service和Dao类，在配置文件中创建对象，在service中注入dao对象





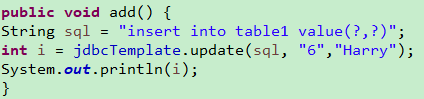
（2）dao中注入jdbcTemplate对象

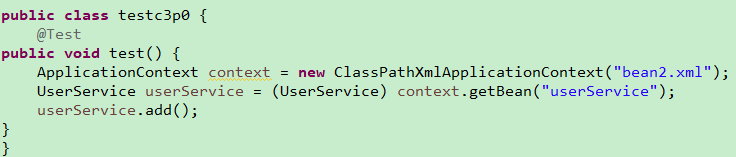


1. jdbcTemplate中注入dataSource对象



1. dao中实现数据库操作





步骤：Service→Dao→jdbcTemplate→dataSource

# Spring的事务管理

事务概念：一组操作，有一个操作不成功，则事务不成功

## 事务特性

不考虑隔离性（多个事务之间不产生影响）产生读问题：脏读，不可重读，虚读，幻读

## 解决读问题

1. 设置隔离级别

## Spring事务管理API

Sping事务管理两种方式

第一种 编程式事务管理（要写代码，一般不用）

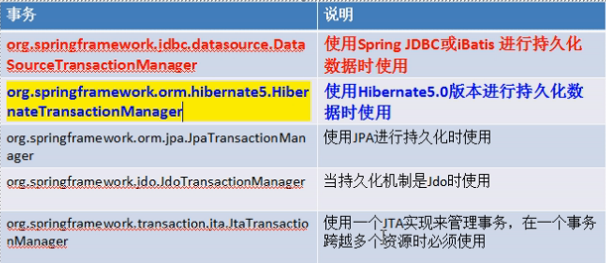
第二种 声明式事务管理

1. 基于Xml配置文件实现
2. 基于注解实现

Spring事务管理的介绍



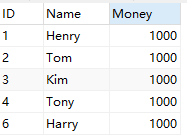
1. spring针对不同的dao层框架，提供接口不同的实现类



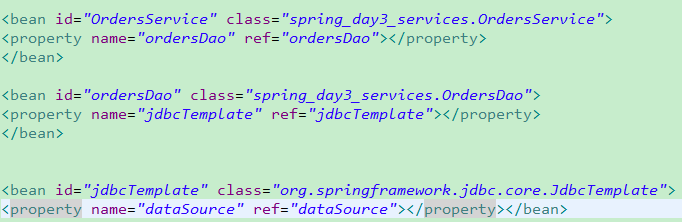
1. 首先：配置事务管理器，创建类的对象

搭建转账环境

1. 创建数据库表，添加数据

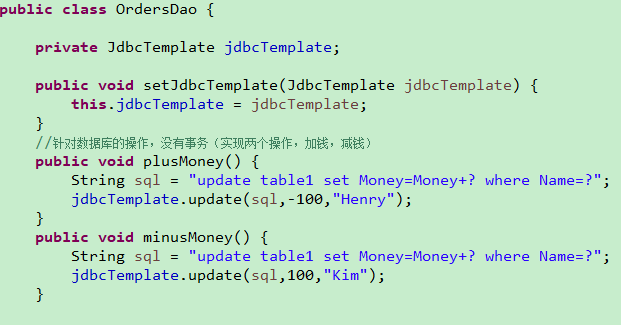


1. 创建Service和dao类，完成注入关系

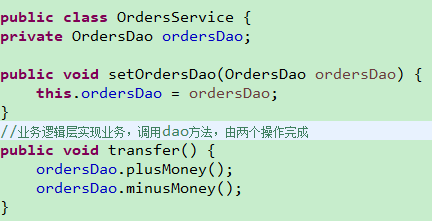


1. service层又叫业务逻辑层（转入，转出两个操作加起来就是业务）
2. dao层，单独对数据库操作层，在dao层不添加业务
3. 需要：转账100 分两步，转出减少100，转入增加100

Dao层实现对数据库操作



Service层实现业务



3**假如产生问题**

如果只扣了100，但是没有转入到户口中

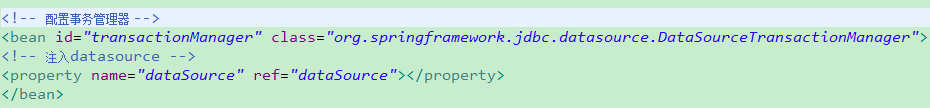
解决方法：

添加事务解决，出现异常进行事务回滚

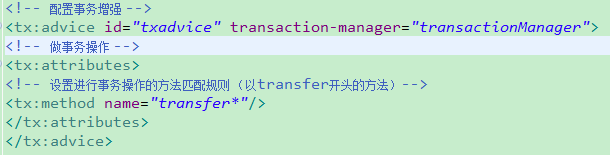
## 声明式事务管理（xml配置）

### 配置文件方式使用aop思想配置

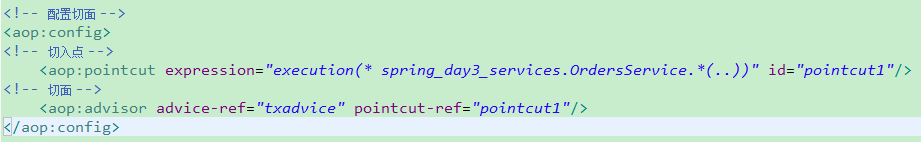
第一步 配置事务管理器



第二步 配置事务增强（把方法作为一个事务，若此方法有异常，则所做的改变不生效，回滚）

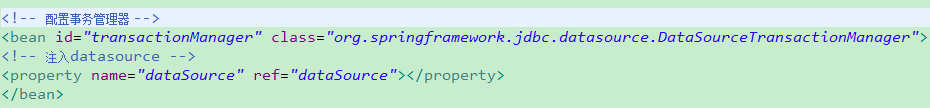


第三部 配置切面



## 声明式事务管理（注解方式）

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注解（所有的方法都会变成事务）

