# Spring

## Spring是什么？

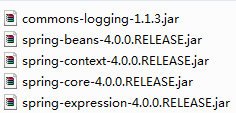
* Spring 是一个开源框架.
* Spring 为简化企业级应用开发而生. 使用 Spring 可以使简单的 JavaBean 实现以前只有 EJB 才能实现的功能.
* Spring 是一个 IOC(DI) 和 AOP 容器框架.

## 安装 SPRING TOOL SUITE

SPRING TOOL SUITE 是一个 Eclipse 插件，利用该插件可以更方便的在 Eclipse 平台上开发基于 Spring 的应用。

## 搭建 Spring 开发环境

* 把以下 jar 包加入到工程的 classpath 下



## 配置bean

配置形式：基于 XML 文件的方式；基于注解的方式

## Java bean和Spring bean区别

* 用处不同：

　　传统javabean更多地作为值传递参数，而spring中的bean用处几乎无处不在，任何组件都可以被称为bean。

* 写法不同：

　　传统javabean作为值对象，要求每个属性都提供getter和setter方法；但spring中的bean只需为接受设值注入的属性提供setter方法。

* 生命周期不同：

　　传统javabean作为值对象传递，不接受任何容器管理其生命周期；spring中的bean有spring管理其生命周期行为

## bean的作用域 （了解）

* singleton：当一个bean的作用域为Singleton，那么Spring IoC容器中只会存在一个共享的bean实例，并且所有对bean的请求，只要id与该bean定义相匹配，则只会返回bean的同一实例。Singleton是单例类型，就是在创建起容器时就同时自动创建了一个bean的对象，不管你是否使用，他都存在了，每次获取到的对象都是同一个对象。注意，Singleton作用域是Spring中的缺省作用域。要在XML中将bean定义成singleton，可以这样配置：

<bean id="ServiceImpl" class="cn.csdn.service.ServiceImpl" scope="singleton">

* Prototype：当一个bean的作用域为Prototype，表示一个bean定义对应多个对象实例。Prototype作用域的bean会导致在每次对该bean请求（将其注入到另一个bean中，或者以程序的方式调用容器的getBean()方法）时都会创建一个新的bean实例。Prototype是原型类型，它在我们创建容器的时候并没有实例化，而是当我们获取bean的时候才会去创建一个对象，而且我们每次获取到的对象都不是同一个对象。根据经验，对有状态的bean应该使用prototype作用域，而对无状态的bean则应该使用singleton作用域。在XML中将bean定义成prototype，可以这样配置：

<bean id="account" class="com.foo.DefaultAccount" scope="prototype"/>

* WEB 环境作用域：request、session和global session三种作用域仅在基于web的应用中使用

## bean的生命周期（了解）

## IOC 和 DI

* IOC(Inversion of Control)：其思想是**反转资源获取的方向**. 传统的资源查找方式要求组件向容器发起请求查找资源. 作为回应, 容器适时的返回资源. 而应用了 IOC 之后, 则是**容器主动地将资源推送给它所管理的组件, 组件所要做的仅是选择一种合适的方式来接受资源**. 这种行为也被称为查找的被动形式。
* DI(Dependency Injection) — IOC 的另一种表述方式：即**组件以一些预先定义好的方式(例如: setter 方法)接受来自如容器的资源注入.** 相对于 IOC 而言，这种表述更直接。

## Spring中实例化bean的方式通过xml方式

## 一 依赖注入：

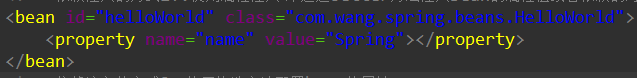
配置bean

class: bean 的全类名通过反射的方式在IOC容器中创建Bean，所以要求Bean中必须一个无参的构造器。

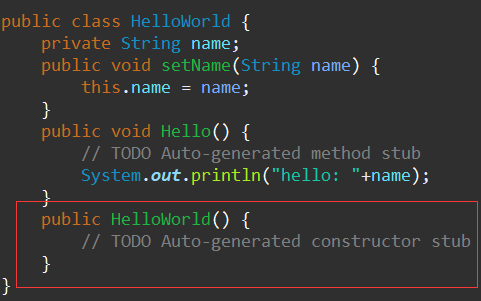
id:标识容器中的bean，id唯一

### 1.使用属性注入

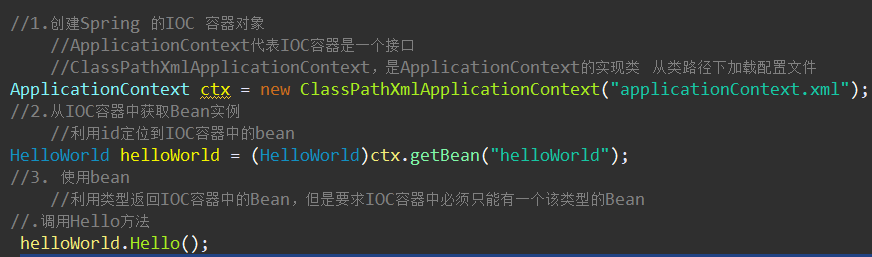
即通过setter方法注入Bean的属性值或者依赖的对象（最常使用）使用property 标签。（value是传入的值，name是类中的变量名）



com.wang.spring.beans包中的HelloWorld类



测试类

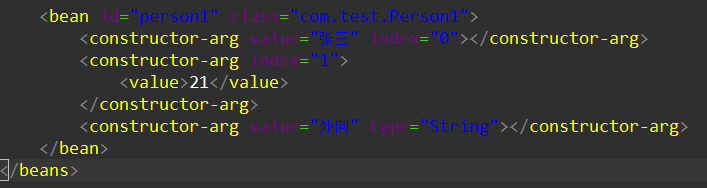


### 2使用构造方法配置bean的属性

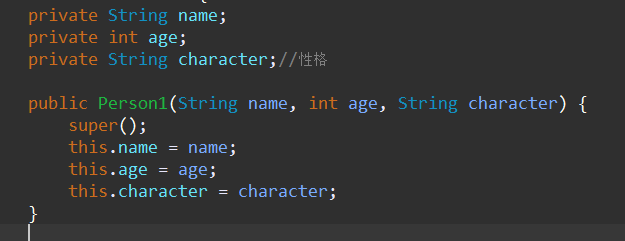
* 实例一

使用构造器注入属性值可以指定参数的位置和参数的类型，以区分重载的构造器使用constructor-arg标签。

对于constructor-arg中的value可以写在标签里面，也可以写在外面应用value标签。index指定有参构造方法中值的顺序，不写默认对应。也可以通过type指定类型。



com.test包中的Person1类。

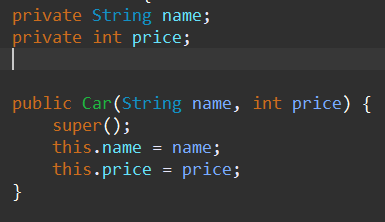


* 实例二

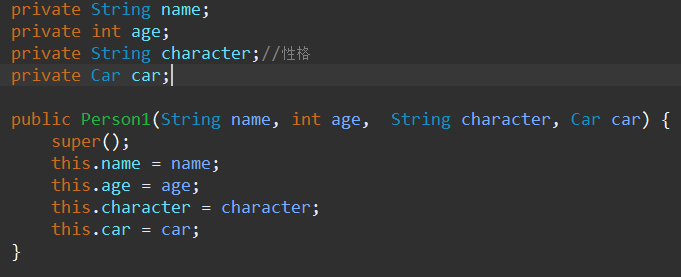
使用内部bean方法test.xml配置文件



Car类



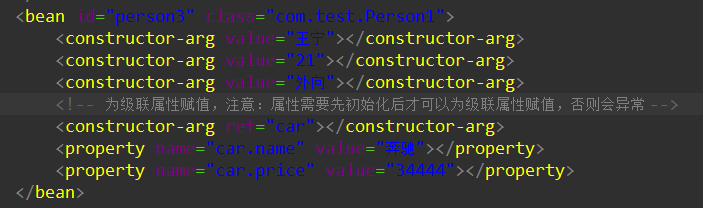
Person1类



使用外部bean方法通过ref属性引用外部方法



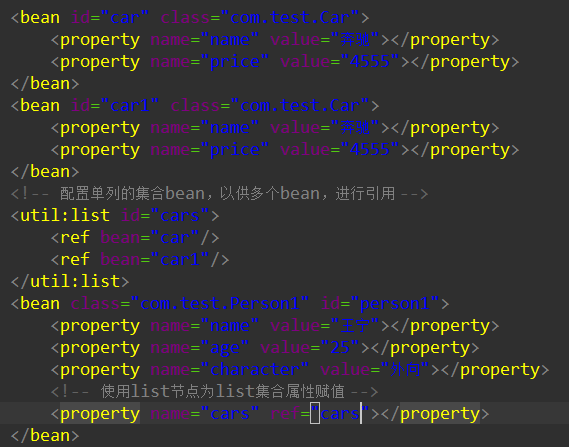
使用构造柱级联属性



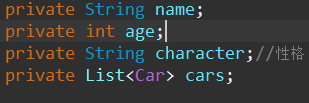
若类中的属性为list集合可使用list节点为list集合属性赋值



也可以配置单独的集合bean，以供多个bean，进行引用



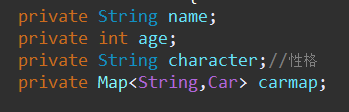
Person1类中的内容为



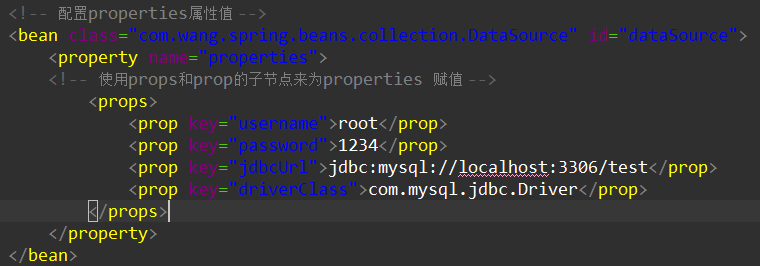
若类中的属性为Mao集合可使用map节点为map集合属性赋值



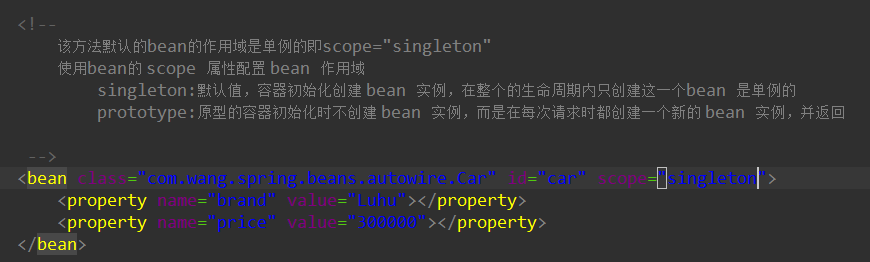
Person1中的属性



配置properties（数据源，java连接数据库使用的）属性值



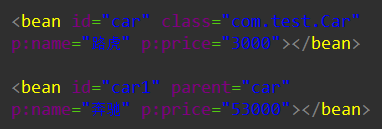
### 3.配置bean的作用域



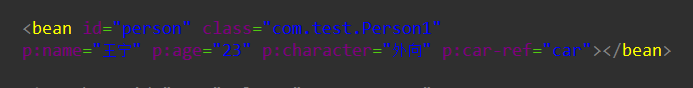
### 4.使用bean中的p命名空间配置bean



Bean的继承配置，使用parent属性指定继承的bean

  
使用变量名的-ref，指定其他bean

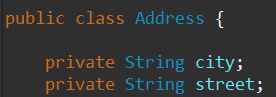
要求在配置Person是必须有一个关联的car 换句话 person 这个bean 依赖与Car 这个bean.

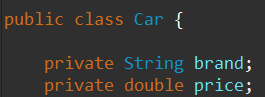


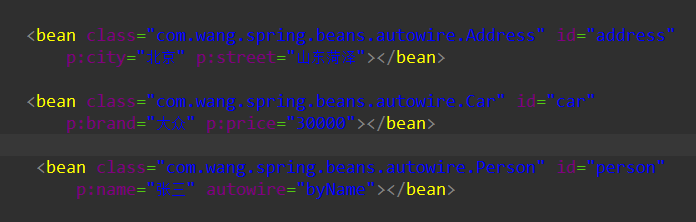
使用autowire属性指定自动装配（实际开发中不常使用）

1、byName 根据bean的名字和当前bean的setter风格的属性名进行自动装配若有匹配的，则自动装配，若没有匹配的则不装配。

2、byType 根据bean的类型和当前bean的属性的类型进行自动装配 。若ioc容器中有一个以上的类型匹配，则抛出异常

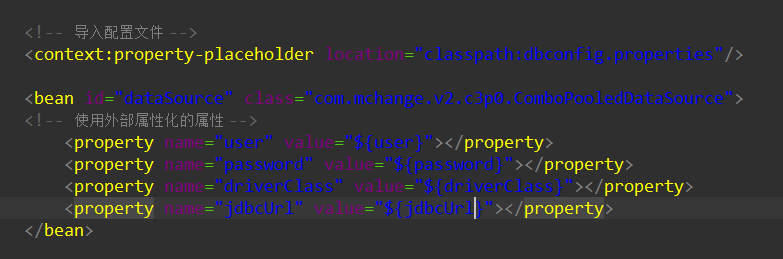




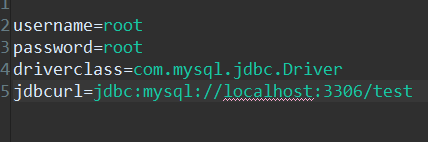


### 5.使用外部文件

通过location指定引用文件路径，${}获取文件中的属性值



dbconfig.properties

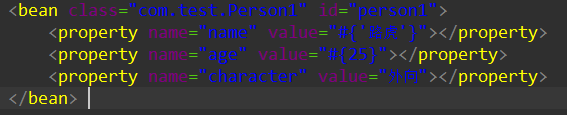


### 6. SpEL表达式

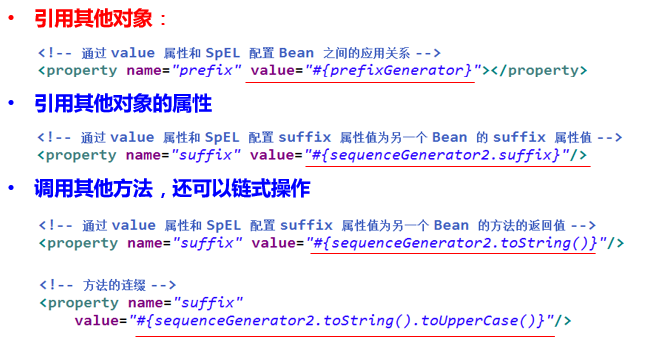
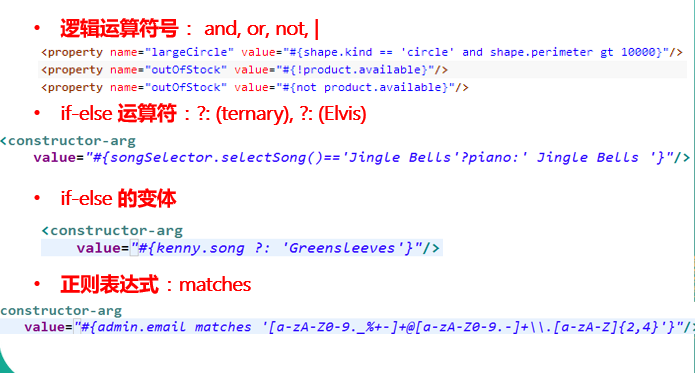
SpEL 使用 #{…} 作为定界符，所有在大框号中的字符都将被认为是 SpEL

* 通过 SpEL 可以实现：
  + 通过 bean 的 id 对 bean 进行引用
  + 调用方法以及引用对象中的属性
  + 计算表达式的值
  + 正则表达式的匹配

对应字String类型：String可以使用单引号或者双引号作为字符串的定界符号





## 二 工程模式

### 静态工厂方法

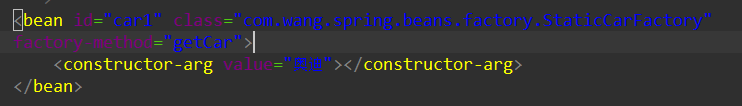
通过静态工厂方法来配置 bean，注意不是配置静态工厂方法实例，而是配置 bean 实例。

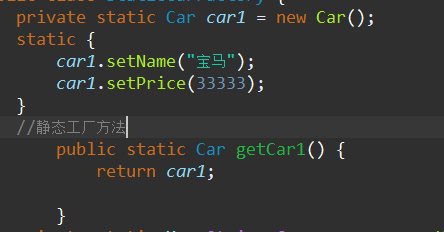
class:指向静态方法的全类名

factory-method:指向静态工厂的方法名

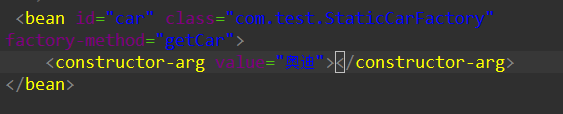
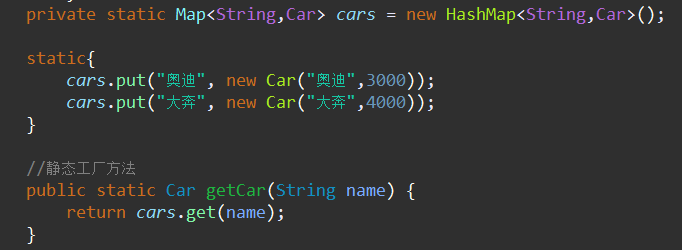
constructor-arg:如果工厂方法需要传入参数，则使用constructor-arg类配置参数

* 无参

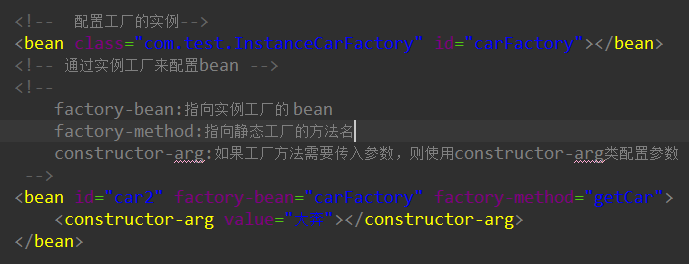




* 有参

### 2．和实例工厂方法





### 3. FactoryBean

## 三 通过注解配置bean

* 组件扫描(component scanning): Spring 能够从 classpath 下自动扫描, 侦测和实例化具有特定注解的组件.
* 特定组件包括:
  + @Component: 基本注解, 标识了一个受 Spring 管理的组件
  + @Respository: 标识持久层组件
  + @Service: 标识服务层(业务层)组件
  + @Controller: 标识表现层组件

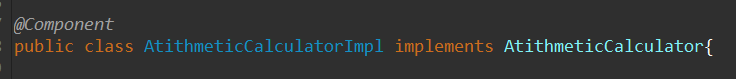


* 对于扫描到的组件, **Spring 有默认的命名策略**: 使用非限定类名, 第一个字母小写**. 也可以在注解中通过 value 属性值标识组件的名称。**

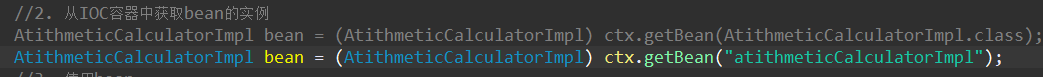
1.配置注解



2.指定类



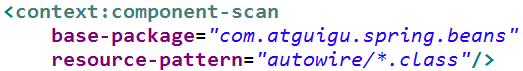
3引用类（使用注解后，引用类时默认第一个字母小写）



当在组件类上使用了特定的注解之后, 还需要在 Spring 的配置文件中声明 <context:component-scan> ：

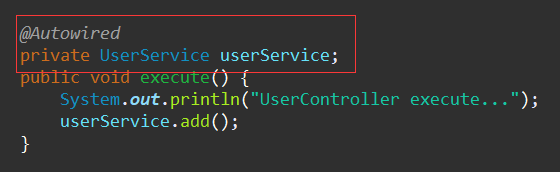
base-package 属性指定一个需要扫描的基类包，Spring 容器将会扫描这个基类包里及其子包中的所有类.

如果仅希望扫描特定的类而非基包下的所有类，可使用 resource-pattern 属性过滤特定的类



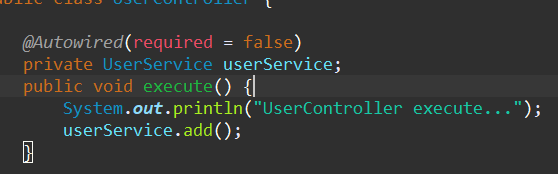
### 使用 @Autowired 自动装配 Bean

@Autowired 可以自动建立bean和bean直接的关系



如果容器中没有UserService

可以设置 @Authwired 注解的 required 属性为 false



@Authwired 注解也可以应用在**数组类型**的属性上, 此时 Spring 将会把所有匹配的 Bean 进行自动装配.

@Authwired 注解也可以应用在**集合属性**上, 此时 Spring 读取该集合的类型信息, 然后自动装配所有与之兼容的 Bean.

@Authwired 注解用**在 java.util.Map** 上时, 若该 Map 的键值为 String, 那么 Spring 将自动装配与之 Map 值类型兼容的 Bean, 此时 Bean 的名称作为键值。

* Spring 还支持 @Resource 和 @Inject 注解，这两个注解和 @Autowired 注解的功用类似。

@Resource 注解要求提供一个 Bean 名称的属性，若该属性为空，则自动采用标注处的变量或方法名作为 Bean 的名称

@Inject 和 @Autowired 注解一样也是按类型匹配注入的 Bean， 但没有 reqired 属性

**建议使用 @Autowired 注解**

## 四 Spring AOP

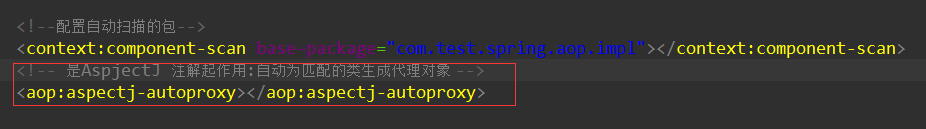
* **AspectJ**：Java 社区里最完整最流行的 AOP 框架.
* 在 Spring2.0 以上版本中, 可以使用基于 AspectJ 注解或基于 XML 配置的 AOP

### 1.基于注解的方法

1. 项目中所需要的jar包



1. 在配置文件中加入aop的命名空间



1. 把横切关注点的代码抽象到切面的类中

* 切面首先是一个 IOC 中的bean 即加入@Component注解
* 切面还需要加入@Aspect 注解

1. 在类中声明各种通知
   1. **@Before:** 前置通知, 在方法执行之前执行
   2. **@After:** 后置通知, 在方法执行之后执行
   3. **@AfterRunning:** 返回通知, 在方法返回结果之后执行
   4. **@AfterThrowing:** 异常通知, 在方法抛出异常之后
   5. **@Around:** 环绕通知, 围绕着方法执行

#### 1.1前置通知@Before

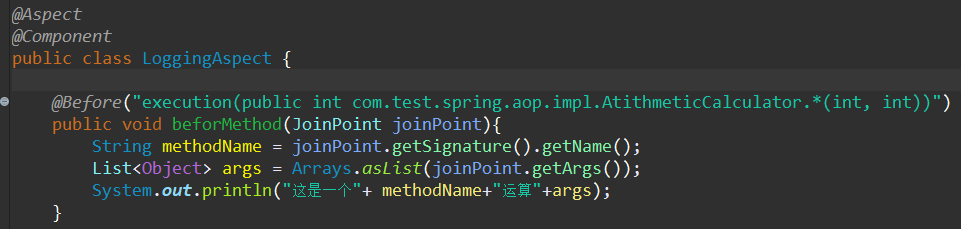
前置通知，在方法执行之前执行

JoinPoint 对象

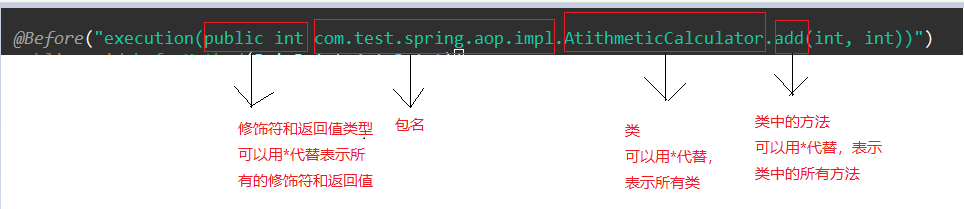
JoinPoint对象封装了SpringAop中切面方法的信息,在切面方法中添加JoinPoint参数,就可以获取到封装了该方法信息的JoinPoint对象.

joinPoint.getSignature().getName()获取方法名

joinPoint.getArgs()获取方法参数

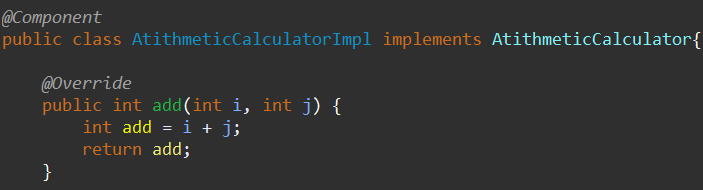


解释





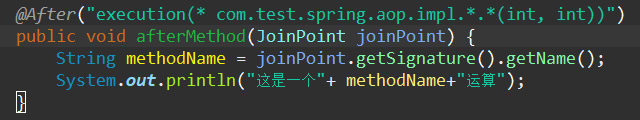
AtithmeticCalculator类



输出结果是，在执行add方法前输出一句话

#### [1.2后置通知@After](mailto:1.2后置通知@After)

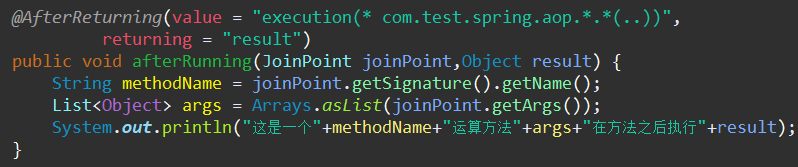
* 后置通知：就是在目标方法执行后（无论是否发生异常），执行的通知
* 后置通知不能返回目标方法执行的结果。



#### 1.3返回通知 @AfterReturning

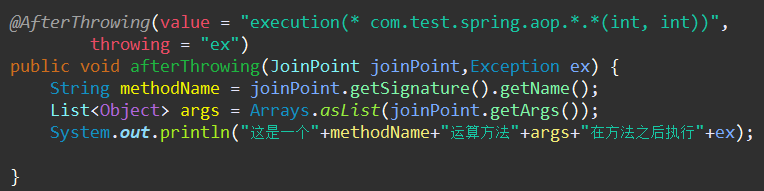
* 返回通知：在方法正常执行结束（发生异常不执行）后执行的通知
* 返回通知是可以返回方法的返回值的

Returning 获取返回值



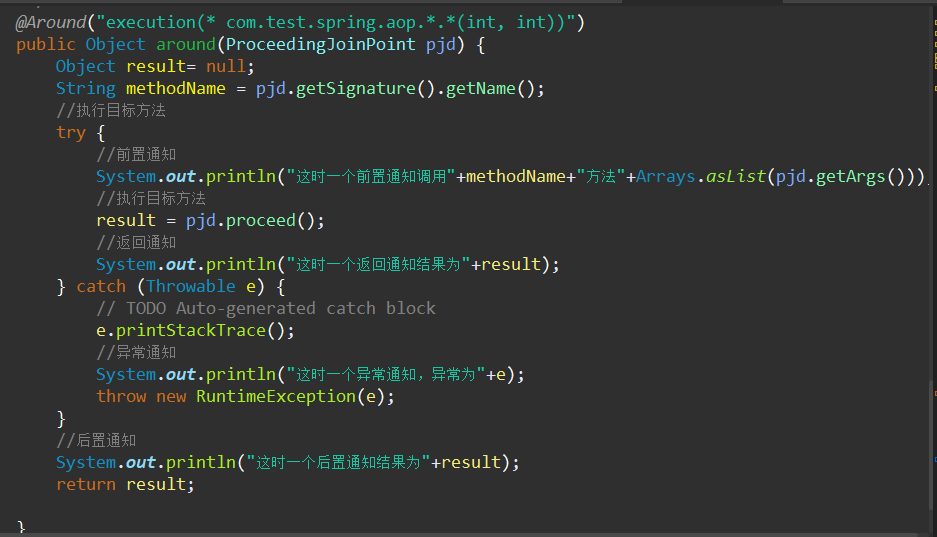
#### 异常通知@AfterThrowing

* 异常通知：在方法执行发生异常时，执行的代码
* 可以访问到异常对象；且可以指定出现特定异常时在执行通知代码



#### [1.5环绕通知@Around](mailto:1.5环绕通知@Around)（了解）

* 环绕通知需要携带ProceedingJoinPoint 类型的参数
* 环绕通知类似代理的全过程 ProceedingJoinPoint 这个类型的参数可以决定是否执行目标方法
* 有返回值，返回值即为目标方法的返回值

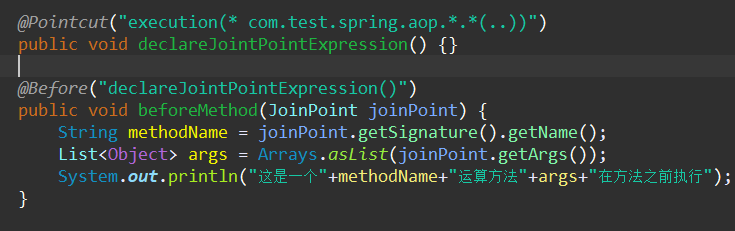


#### 1.6可以使用@Order指定切面的优先级，值越小优先级越高



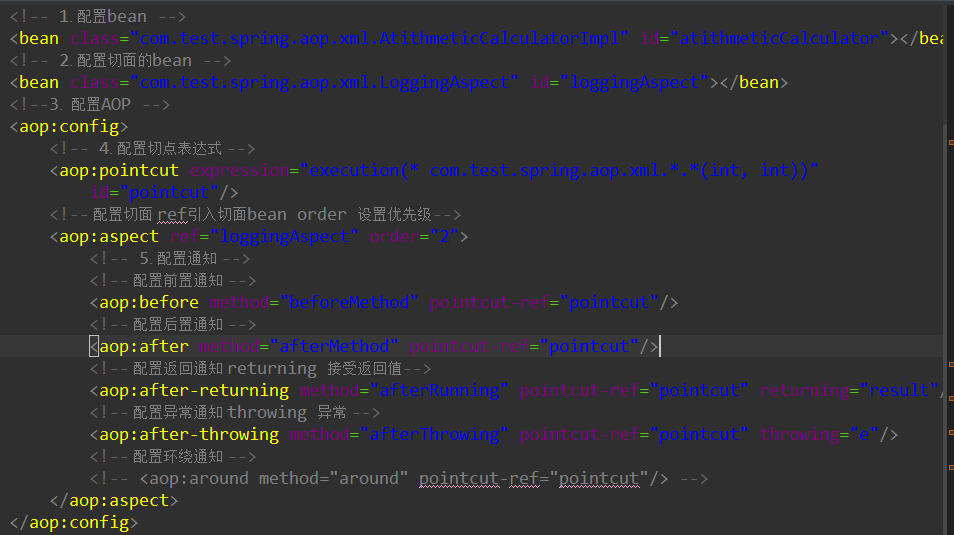
#### 1.7重用切点表达式

* 定义一个方法，用于声明切入点表达式，一般该方法中不需要填入其他代码
* 使用@Pointcut 来声明切入点表达式，
* 后面的其他通知直接使用方法名来引入切点表达式
* 不同类中的可以可以使用，设置切入点的类全名点方法名使用



### 基于配置文件方式

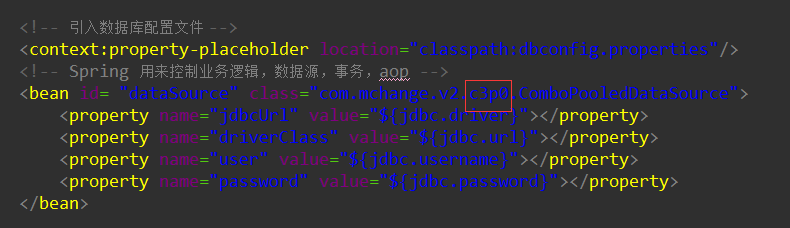
配置文件applicationContext-xml.xml



AOP切面类



## 五 Spring地JDBC的支持



## 六 Spring 中的事务管理

* 事务管理是企业级应用程序开发中必不可少的技术, **用来确保数据的完整性和一致性**.
* 事务就是一系列的动作, 它们被当做一个单独的工作单元. 这些动作要么全部完成, 要么全部不起作用。

事务就是对一系列的数据库操作进行统一的提交或回滚操作，比如说做一个转账功能，要更改帐户两边的数据，这时候就必须要用事务才能算是严谨的做法。要么成功，要么失败，保持数据一致性。如果中间有一个操作出现异常，那么回滚之前的所有操作。

### 事务的四个关键属性(ACID)

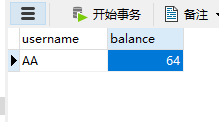
* 1. 原子性(atomicity): 事务是一个原子操作, 由一系列动作组成. 事务的原子性确保动作要么全部完成要么完全不起作用（例如转账，A给B转账，A的账号减500，B的账号加500）.
  2. 一致性(consistency): 一旦所有事务动作完成, 事务就被提交. 数据和资源就处于一种满足业务规则的一致性状态中.
  3. 隔离性(isolation): 可能有许多事务会同时处理相同的数据, 因此每个事物都应该与其他事务隔离开来, 防止数据损坏.
  4. 持久性(durability): 一旦事务完成, 无论发生什么系统错误, 它的结果都不应该受到影响. 通常情况下, 事务的结果被写到持久化存储器中.

### 事务管理的两种方式

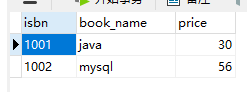
* **编程式事务管理: 将事务管理代码嵌入到业务方法中来控制事务的提交和回滚**. 在编程式管理事务时, 必须在每个事务操作中包含额外的事务管理代码.
* **声明式事务管理**: 大多数情况下比编程式事务管理更好用. 它**将事务管理代码从业务方法中分离出来, 以声明的方式来实现事务管理**. 事务管理作为一种横切关注点, 可以通过 AOP 方法模块化. **Spring 通过 Spring AOP 框架支持声明式事务管理**.

### 声明式事务

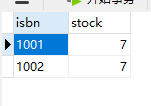
account



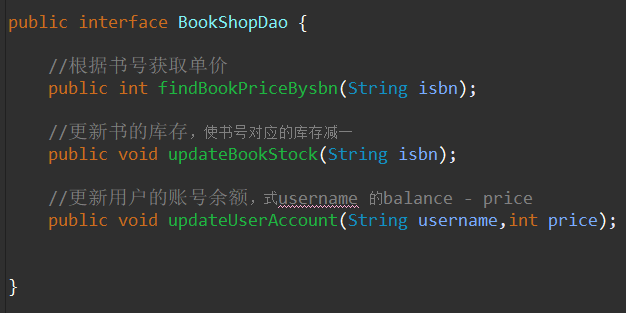
book



book\_stock



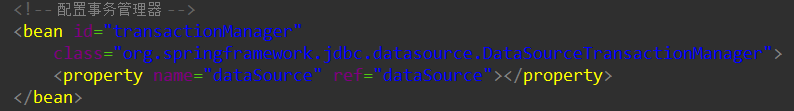
BookshopDao接口



BookshopDao接口的实现类BookShopDaoImpl



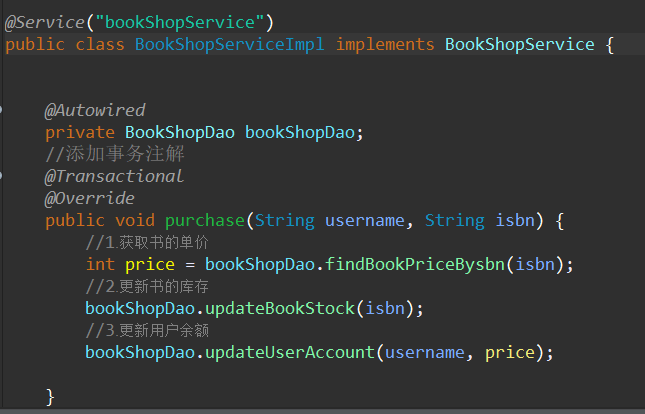
1.配置事务管理器



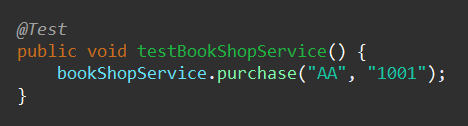
2. 启用事务注解



3. 添加事务注解



测试类



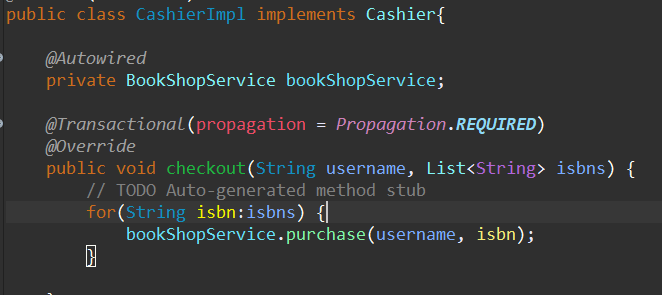
用户AA要买版本号为 1001的书，如果用户的余额或者，书的库存不够，由于添加事务会发生回滚，交易失败。

### 4.事务的传播行为

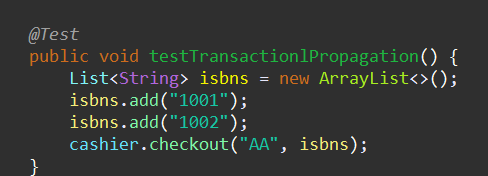
4.1默认的传播行为就是 REQUIRED

checkout 方法中调用的另一个带有事务的方法purchase

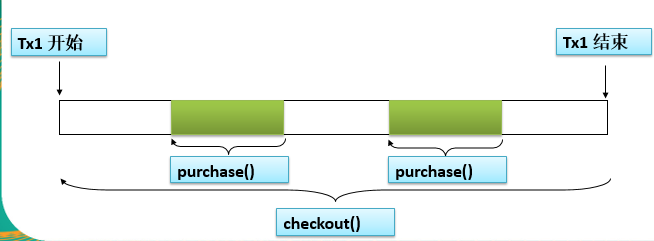
* 使用propagation 指定事务的传播行为，即当前的事务方法被另一个事务的方法调用时
* 如果使用事务，默认取值为REQUIRED，只写@Transactional等同于下图写法。即使用的式调用方法的事务（purchase的事务）。



测试类，一个人同时买两本书。

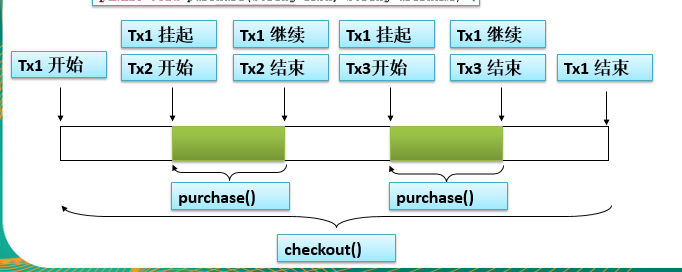


只有有一个条件不满足（如果有一本书库存不足，或者余额不够两本书的钱），就会回滚到原始状态交易失败。



4.2 REQUIRED\_NEW

REQUIRED\_NEW:事务自己的事务（checkout），调用的事务方法的事务被挂起。



一个人买两本书，只有一个条件满足（如果有一本书库存不足，或者余额只够一本书的钱）。只会发生满足条件的书交易。

### 5.事务的隔离级别

事务的隔离级别要得到底层数据库引擎的支持, 而不是应用程序或者框架的支持.

Oracle 支持的 2 种事务隔离级别：READ\_COMMITED , SERIALIZABLE

Mysql 支持 4 中事务隔离级别.

使用isolation 指定事务的隔离级别，最常用的取值为READ\_COMMITTED

### 6.事务的回滚属性

默认情况下Spring的声明式事务对所有的运行时异常进行回滚，也可以通过对应的属性进行设置，通常情况下默认即可。

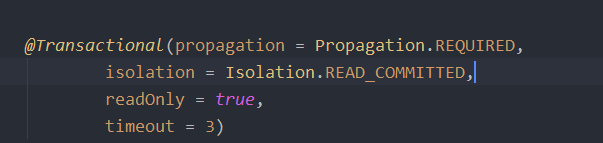
### 7.事务是只读属性

使用readOnly 指定事务是否只读

**只读事务属性**: 表示这个事务只读取数据但不更新数据, 这样可以帮助数据库引擎优化事务.应该设置readOnly=true

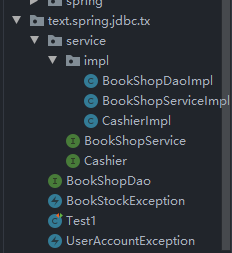
### 8.事务的占用时间

timmout指定强制回滚之前事务占用的时间

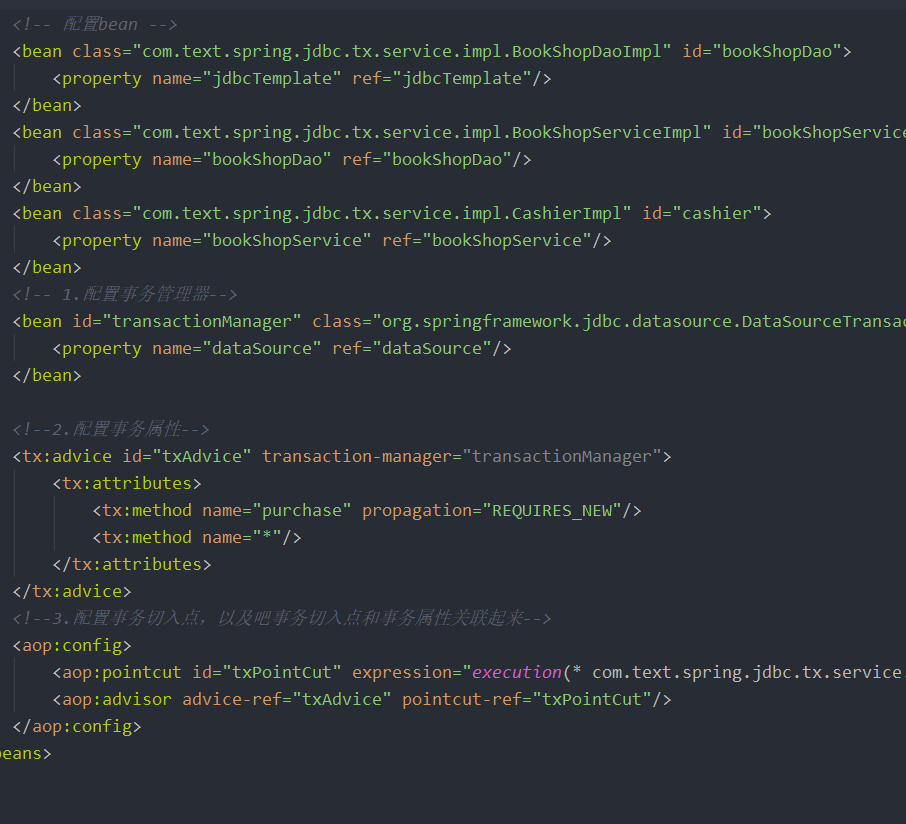


### 9.基于xml的配置方式

类的结构

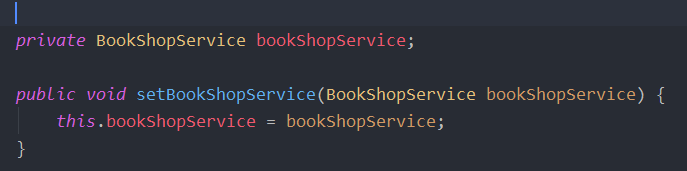


Spring配置文件

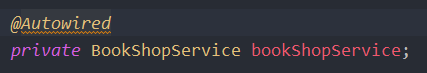


* 配置bean。是将类中的成员变量引入相当于注解中的@Autowired

使用xml方式



使用注解方式



* DataSourceTransactionManager为事务管理器
* tx:advice和tx:attributes配合使用，用tx:method 来指定事务的行为
* aop:pointcut配置事务的切入点 expression 用来指定方法

1、execution(): 表达式主体 (必须加上execution)。

 2、第一个\*号：表示返回值类型，\*号表示所有的类型。

 3、包名：表示需要拦截的包名，后面的两个句点表示当前包和当前包的所有子包，cn.smd.service.impl包、子孙包下所有类的方法。

 4、第二个\*号：表示类名，\*号表示所有的类。

 5、\*(..):最后这个星号表示方法名，\*号表示所有的方法，后面括弧里面表示方法的参数，两个句点表示任何参数。

* <aop:advisor advice-ref="txAdvice" pointcut-ref="txPointCut"/> 把切入点和事务属性关联起来。