# Spring介绍

## Spring 优点

1. 降低了组件之间的耦合性 ，实现了软件各层之间的解耦
2. **可以使用容易提供的众多服务，如事务管理，消息服务等**
3. **容器提供单例模式支持**
4. **容器提供了AOP技术，利用它很容易实现如权限拦截，运行期监控等功能**
5. **容器提供了众多的辅助类，能加快应用的开发**
6. **spring对于主流的应用框架提供了集成支持，如hibernate，JPA，Struts等**
7. **spring属于低侵入式设计，代码的污染极低**
8. **独立于各种应用服务器**
9. **spring的DI机制降低了业务对象替换的复杂性**
10. **Spring的高度开放性，并不强制应用完全依赖于Spring，开发者可以自由选择spring的部分或全部**

## Spring 的理解

通过依赖注入机制降低了编码侵入性，降低了对象引用的复杂性；

通过AOP编程降低了程序设计的耦合性；

更容易使用提供的很多服务组件，如事务管理，消息服务等

## Spring Bean生命周期

# IOC机制

定位：资源配置import、classpath、url

加载：解析配置文件，把bean包装成BeanDefinition对象

注册：把已经初始化的BeanDefinition放入到IOC容器中（Map）

# DI机制

依赖注入（Dependecy Injection）和控制反转（Inversion of Control）是同一个概念，具体的讲：当某个角色 需要另外一个角色协助的时候，在传统的程序设计过程中，通常由调用者来创建被调用者的实例。但在spring中 创建被调用者的工作不再由调用者来完成，因此称为控制反转。创建被调用者的工作由spring来完成，然后注入调用者，因此也称为依赖注入。   
spring以动态灵活的方式来管理对象，注入的两种方式，设置注入和构造注入。设置注入的优点：直观，自然   
构造注入的优点：可以在构造器中决定依赖关系的顺序。

***依赖注入的时间：***

(1).用户第一次通过getBean 方法向IOC 容索要Bean 时，IOC 容器触发依赖注入。

(2).当用户在Bean 定义资源中为<bean>元素配置了lazy-init 属性，即让容器在解析注册Bean 定义时进行预实例化，触发依赖注入。

Spring中的对象默认是单例的 scope-singlton；

# AOP

面向切面编程（AOP）完善spring的依赖注入（DI），面向切面编程在spring中主要表现为两个方面   
1.面向切面编程提供声明式事务管理   
2.spring支持用户自定义的切面   
  
面向切面编程（aop）是对面向对象编程（oop）的补充，   
面向对象编程将程序分解成各个层次的对象，面向切面编程将程序运行过程分解成各个切面。   
AOP从程序运行角度考虑程序的结构，提取业务处理过程的切面，oop是静态的抽象，aop是动态的抽象，   
是对应用执行过程中的步骤进行抽象，，从而获得步骤之间的逻辑划分。   
  
aop框架具有的两个特征：   
1.各个步骤之间的良好隔离性   
2.源代码无关性

## [Spring AOP 实现原理](http://blog.csdn.net/moreevan/article/details/11977115)

实现AOP的技术，主要分为两大类：

一是采用动态代理技术，利用截取消息的方式，对该消息进行装饰，以取代原有对象行为的执行；

二是采用静态织入的方式，引入特定的语法创建“方面”，从而使得编译器可以在编译期间织入有关“方面”的代码。

## AOP使用场景

AOP用来封装横切关注点，具体可以在下面的场景中使用:

Authentication 权限

Caching 缓存

Context passing 内容传递

Error handling 错误处理

Lazy loading　懒加载

Debugging　　调试

logging, tracing, profiling and monitoring　记录跟踪　优化　校准

Performance optimization　性能优化

Persistence　　持久化

Resource pooling　资源池

Synchronization　同步

Transactions 事务

## AOP相关概念

方面（Aspect）：一个关注点的模块化，这个关注点实现可能另外横切多个对象。事务管理是J2EE应用中一个很好的横切关注点例子。方面用**[spring](http://lib.csdn.net/base/javaee" \t "_blank" \o "Java EE知识库)**的 Advisor或拦截器实现。

连接点（Joinpoint）: 程序执行过程中明确的点，如方法的调用或特定的异常被抛出。

通知（Advice）: 在特定的连接点，AOP框架执行的动作。各种类型的通知包括“around”、“before”和“throws”通知。通知类型将在下面讨论。许多AOP框架包括Spring都是以拦截器做通知模型，维护一个“围绕”连接点的拦截器链。Spring中定义了四个advice: BeforeAdvice, AfterAdvice, ThrowAdvice和DynamicIntroductionAdvice

切入点（Pointcut）: 指定一个通知将被引发的一系列连接点的集合。AOP框架必须允许开发者指定切入点：例如，使用正则表达式。 Spring定义了Pointcut接口，用来组合MethodMatcher和ClassFilter，可以通过名字很清楚的理解， MethodMatcher是用来检查目标类的方法是否可以被应用此通知，而ClassFilter是用来检查Pointcut是否应该应用到目标类上

引入（Introduction）: 添加方法或字段到被通知的类。 Spring允许引入新的接口到任何被通知的对象。例如，你可以使用一个引入使任何对象实现 IsModified接口，来简化缓存。Spring中要使用Introduction, 可有通过DelegatingIntroductionInterceptor来实现通知，通过DefaultIntroductionAdvisor来配置Advice和代理类要实现的接口

目标对象（Target Object）: 包含连接点的对象。也被称作被通知或被代理对象。POJO

AOP代理（AOP Proxy）: AOP框架创建的对象，包含通知。 在Spring中，AOP代理可以是JDK动态代理或者CGLIB代理。

织入（Weaving）: 组装方面来创建一个被通知对象。这可以在编译时完成（例如使用AspectJ编译器），也可以在运行时完成。Spring和其他纯**[Java](http://lib.csdn.net/base/java" \t "_blank" \o "Java 知识库)**AOP框架一样，在运行时完成织入。

## AOP的配置方式

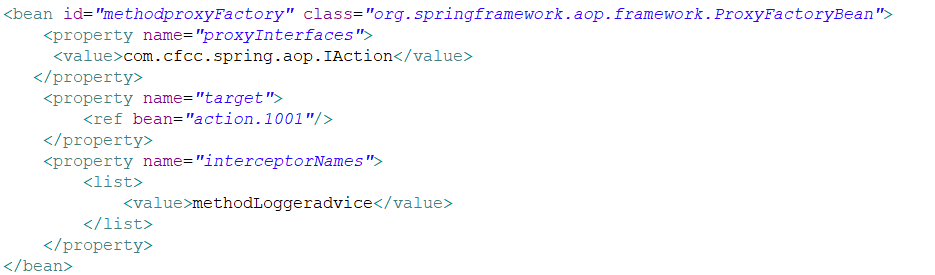
### ProxyFactoryBean

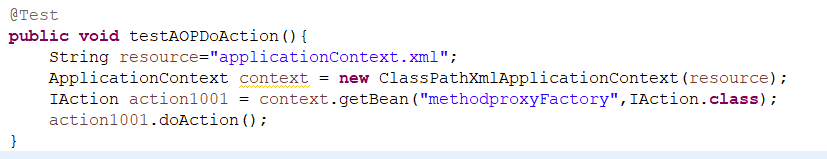
创建基于Spring Bean 的AOP代理

（1）MethodInterceptors 和Advisors在当前的Bean Factory中被一个bean name列表表示，这个列表通过“**interceptorNames**”属性指定。

可以通过匹配符进行匹配，如：global\* 代表以global开头的所有interceptors

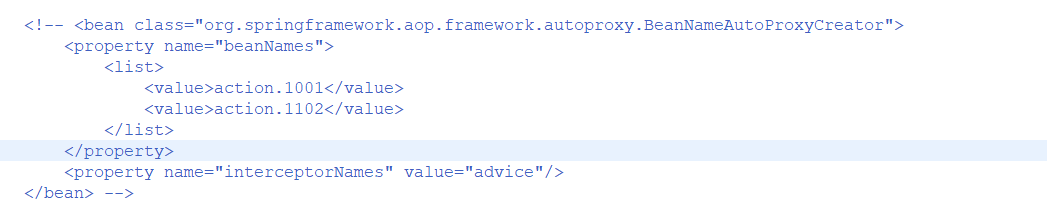
（2）当proxy interfaces指定时创建一个JDK代理，如果不指定则为目标创建一个CGLIB代理

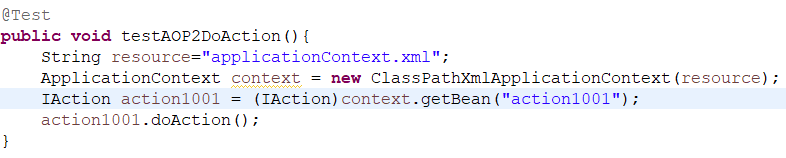




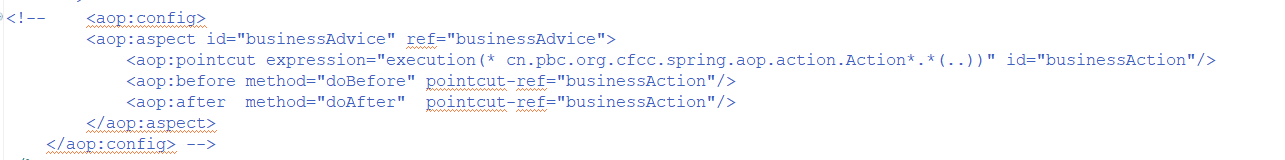
### BeanNameAutoProxyCreator

自动代理创建器，它通过名称列表标识bean。可以使用匹配符"xxx\*", and "\*xxx" matches.





### Aspect

****

### 区别

因为ProxyFactoryBean 不支持代理列表，对于每个业务类我们都要配一个ProxyFactoryBean

BeanNameAutoProxyCreator来自动为满足通配条件的bean生产代理，而且在依赖业务类的地方直接使用原始beanName即可

# Spring的事务

## 介绍

**Spring配置声明式事务：**

**\* 配置DataSource**

**\* 配置事务管理器**

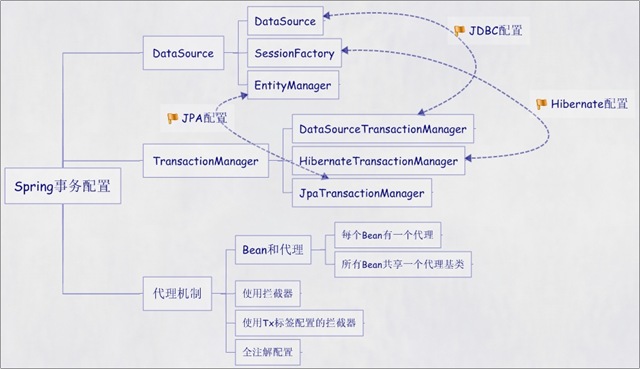
**\* 事务的传播特性**

**\* 那些类那些方法使用事务**

**Spring配置文件中关于事务配置总是由三个组成部分，分别是DataSource、TransactionManager和代理机制这三部分，无论哪种配置方式，一般变化的只是代理机制这部分。**

**DataSource、TransactionManager这两部分只是会根据数据访问方式有所变化，**

**比如使用hibernate进行数据访问 时，DataSource实际为SessionFactory，TransactionManager的实现为 HibernateTransactionManager。**



## Spring事务的配置方式

根据代理机制的不同，Spring事务的配置又有几种不同的方式：

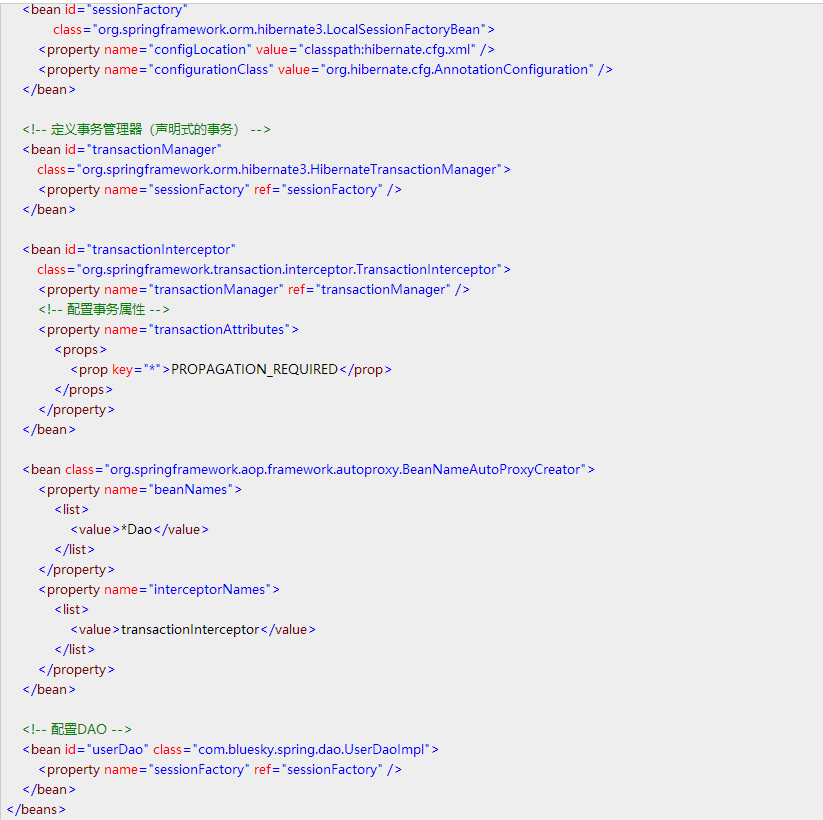
### 每个Bean都有一个代理



### 所有Bean共享一个代理基类



### 使用拦截器

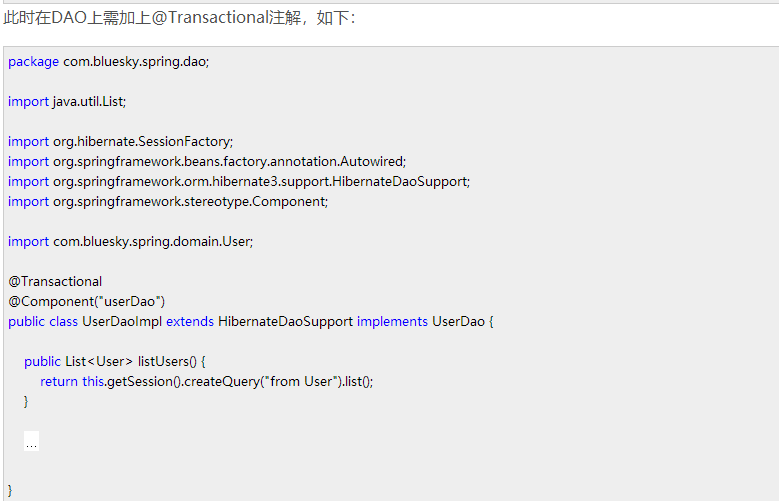


### 使用tx标签配置的拦截器



### 全注解

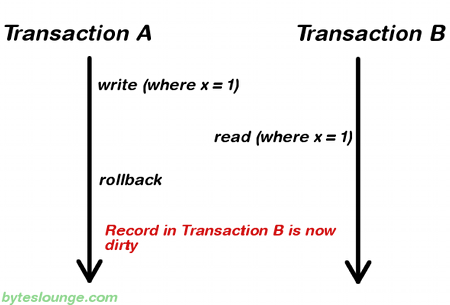




## Spring事务隔离级别

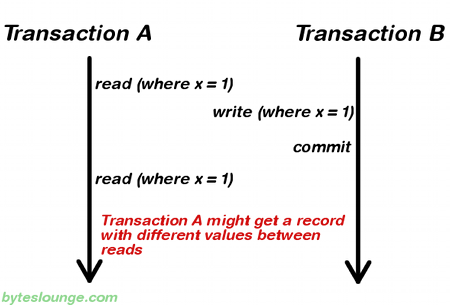
### READ\_UNCOMMITTED

一个事务可以读取到另一个事务未提交的事务记录。出现脏读、还可能出现non-repeatable read(不可重复读)和phantom read(幻读)的问题



### READ\_COMMITTED

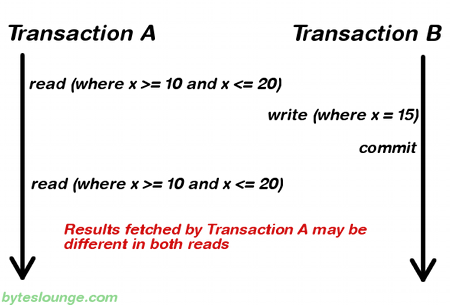
一个事务只能读取到已经提交的记录，不能读取到未提交的记录。在事务A两次读取的过程之间，事务B修改了那条记录并进行提交。因此，事务A前后两次读取的记录不一致。这个问题称为non-repeatable read(不可重复读)。（两次读取的记录不一致，重复读取就会发现问题。) READ\_COMMITTED还可能发生phantom read的问题。



### REPEATABLE\_READ

REPEATABLE\_READ意思是，一个事务可以多次从数据库读取某条记录，而且多次读取的那条记录都是一致的，相同的。这个隔离级别可以避免dirty read和non-repeatable read的问题，但可能发生phantom read的问题。如下图。

事务A两次从数据库读取一系列记录，期间，事务B插入了某条记录并提交。事务A第二次读取时，会读取到事务B刚刚插入的那条记录。在事务期间，事务A两次读取的一系列记录不一致，这个问题称为phantom read。



### SERIALIZABLE

SERIALIZABLE是Spring最强的隔离级别。事务执行时，会在所有级别上加锁，比如read和write时都会加锁，仿佛事务是以串行的方式进行的，而不是一起发生的。这会防止dirty read、non-repeatable read和phantom read的出现，但是，会带来性能的下降。

## Spring事务的传播属性

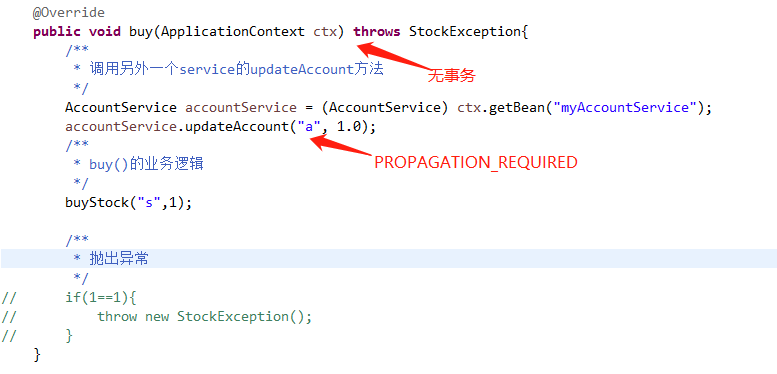
不管是ebj还是spring，我都遇到这样问题，嵌套的事务里的方法都在一个类里的话，  
里面的即使是新建的事务，执行完方法也不会提交。我也没去具体研究什么原因。  
但是我是通过这个线程的转换完成的。

同一个业务类里面 ， 即使声明为 Propagation.REQUIRES\_NEW也不会新启一个事务。  
必须调用另一个类的Propagation.REQUIRES\_NEW方法才行， 这个问题很困惑，至今也不知道是什么原因。

跨Service调用方法时，都会经过org.springframework.aop.framework.CglibAopProxy.DynamicAdvisedInterceptor.intercept()方法，只有经过此处，才能对事务进行控制。

### PROPAGATION\_REQUIRED

如果有事务，那么加入事务，没有的话就起一个新的事务



14876 [main] DEBUG org.springframework.jdbc.datasource.DataSourceTransactionManager - ***Creating new transaction with name [cn.pbc.org.cfcc.spring.transaction.AccountServiceImpl.updateAccount***]: PROPAGATION\_REQUIRED,ISOLATION\_DEFAULT

15541 [main] DEBUG org.springframework.jdbc.datasource.DataSourceTransactionManager - Acquired Connection [1337189208, URL=jdbc:mysql://192.168.168.128:3306/spring, UserName=root@192.168.168.1, MySQL Connector Java] for JDBC transaction

15556 [main] DEBUG org.springframework.jdbc.datasource.DataSourceTransactionManager - Switching JDBC Connection [1337189208, URL=jdbc:mysql://192.168.168.128:3306/spring, UserName=root@192.168.168.1, MySQL Connector Java] to manual commit

15563 [main] DEBUG org.springframework.jdbc.core.JdbcTemplate - Executing prepared SQL update

15564 [main] DEBUG org.springframework.jdbc.core.JdbcTemplate - **E*xecuting prepared SQL statement [update account set balance=balance-? where aname=?]***

15601 [main] DEBUG org.springframework.jdbc.core.JdbcTemplate - SQL update affected 1 rows

15630 [main] DEBUG org.springframework.jdbc.datasource.DataSourceTransactionManager - Initiating transaction commit

15631 [main] DEBUG org.springframework.jdbc.datasource.DataSourceTransactionManager - ***Committing JDBC transaction on Connection*** [1337189208, URL=jdbc:mysql://192.168.168.128:3306/spring, UserName=root@192.168.168.1, MySQL Connector Java]

15634 [main] DEBUG org.springframework.jdbc.datasource.DataSourceTransactionManager - ***Releasing JDBC Connection*** [1337189208, URL=jdbc:mysql://192.168.168.128:3306/spring, UserName=root@192.168.168.1, MySQL Connector Java] after transaction

15634 [main] DEBUG org.springframework.jdbc.datasource.DataSourceUtils - Returning JDBC Connection to DataSource

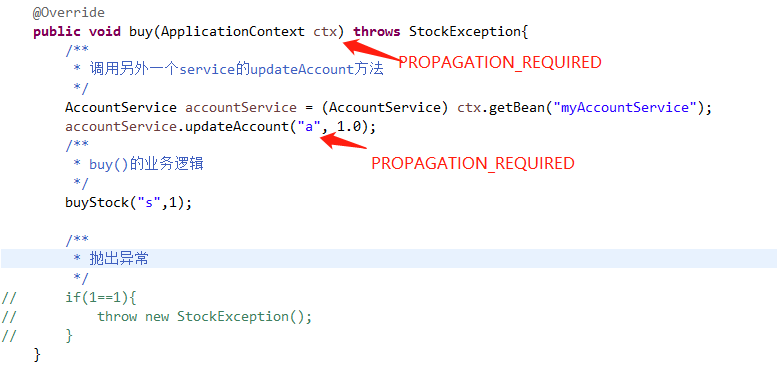
45261 [main] DEBUG org.springframework.jdbc.core.JdbcTemplate - Executing prepared SQL update

45262 [main] DEBUG org.springframework.jdbc.core.JdbcTemplate - ***Executing prepared SQL statement [update stock set count=count+? where sname=?]***

45262 [main] DEBUG org.springframework.jdbc.datasource.DataSourceUtils - Fetching JDBC Connection from DataSource

45355 [main] DEBUG org.springframework.jdbc.core.JdbcTemplate - SQL update affected 1 rows

45355 [main] DEBUG org.springframework.jdbc.datasource.DataSourceUtils - Returning JDBC Connection to DataSource



469 [main] DEBUG org.springframework.jdbc.datasource.DataSourceTransactionManager - ***Creating new transaction with name [cn.pbc.org.cfcc.spring.transaction.StockServiceImpl.buy]:*** PROPAGATION\_REQUIRED,ISOLATION\_DEFAULT,-cn.pbc.org.cfcc.spring.transaction.StockException

870 [main] DEBUG org.springframework.jdbc.datasource.DataSourceTransactionManager - Acquired Connection [633343528, URL=jdbc:mysql://192.168.168.128:3306/spring, UserName=root@192.168.168.1, MySQL Connector Java] for JDBC transaction

874 [main] DEBUG org.springframework.jdbc.datasource.DataSourceTransactionManager - Switching JDBC Connection [633343528, URL=jdbc:mysql://192.168.168.128:3306/spring, UserName=root@192.168.168.1, MySQL Connector Java] to manual commit

875 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory - Returning cached instance of singleton bean 'myAccountService'

875 [main] DEBUG org.springframework.jdbc.datasource.DataSourceTransactionManager - ***Participating in existing transaction***

876 [main] DEBUG org.springframework.jdbc.core.JdbcTemplate - Executing prepared SQL update

877 [main] DEBUG org.springframework.jdbc.core.JdbcTemplate - ***Executing prepared SQL statement [update account set balance=balance-? where aname=?]***

890 [main] DEBUG org.springframework.jdbc.core.JdbcTemplate - SQL update affected 1 rows

904 [main] DEBUG org.springframework.jdbc.core.JdbcTemplate - Executing prepared SQL update

904 [main] DEBUG org.springframework.jdbc.core.JdbcTemplate - ***Executing prepared SQL statement [update stock set count=count+? where sname=?]***

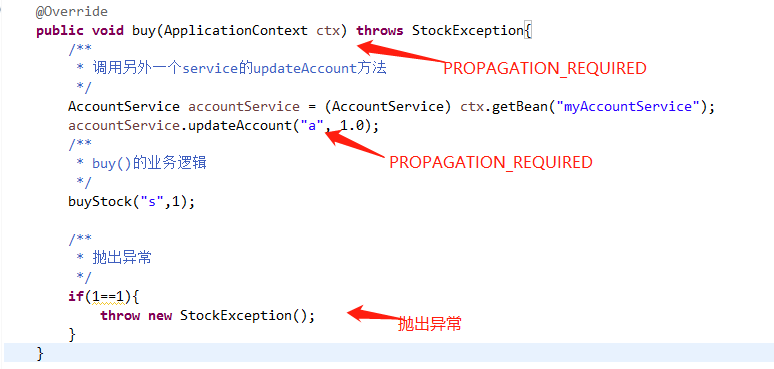
905 [main] DEBUG org.springframework.jdbc.core.JdbcTemplate - SQL update affected 1 rows

906 [main] DEBUG org.springframework.jdbc.datasource.DataSourceTransactionManager - Initiating transaction commit

906 [main] DEBUG org.springframework.jdbc.datasource.DataSourceTransactionManager - ***Committing JDBC transaction on Connection*** [633343528, URL=jdbc:mysql://192.168.168.128:3306/spring, UserName=root@192.168.168.1, MySQL Connector Java]

909 [main] DEBUG org.springframework.jdbc.datasource.DataSourceTransactionManager - ***Releasing JDBC Connection*** [633343528, URL=jdbc:mysql://192.168.168.128:3306/spring, UserName=root@192.168.168.1, MySQL Connector Java] after transaction

909 [main] DEBUG org.springframework.jdbc.datasource.DataSourceUtils - Returning JDBC Connection to DataSource



551 [main] DEBUG org.springframework.jdbc.datasource.DataSourceTransactionManager - ***Creating new transaction with name [cn.pbc.org.cfcc.spring.transaction.StockServiceImpl.buy]***: PROPAGATION\_REQUIRED,ISOLATION\_DEFAULT,-cn.pbc.org.cfcc.spring.transaction.StockException

904 [main] DEBUG org.springframework.jdbc.datasource.DataSourceTransactionManager - Acquired Connection [270088606, URL=jdbc:mysql://192.168.168.128:3306/spring, UserName=root@192.168.168.1, MySQL Connector Java] for JDBC transaction

910 [main] DEBUG org.springframework.jdbc.datasource.DataSourceTransactionManager - Switching JDBC Connection [270088606, URL=jdbc:mysql://192.168.168.128:3306/spring, UserName=root@192.168.168.1, MySQL Connector Java] to manual commit

18251 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory - Returning cached instance of singleton bean 'myAccountService'

24114 [main] DEBUG org.springframework.jdbc.datasource.DataSourceTransactionManager - ***Participating in existing transaction***

24124 [main] DEBUG org.springframework.jdbc.core.JdbcTemplate - Executing prepared SQL update

24126 [main] DEBUG org.springframework.jdbc.core.JdbcTemplate - ***Executing prepared SQL statement [update account set balance=balance-? where aname=?]***

24150 [main] DEBUG org.springframework.jdbc.core.JdbcTemplate - SQL update affected 1 rows

52684 [main] DEBUG org.springframework.jdbc.core.JdbcTemplate - Executing prepared SQL update

52684 [main] DEBUG org.springframework.jdbc.core.JdbcTemplate - ***Executing prepared SQL statement [update stock set count=count+? where sname=?]***

52687 [main] DEBUG org.springframework.jdbc.core.JdbcTemplate - SQL update affected 1 rows

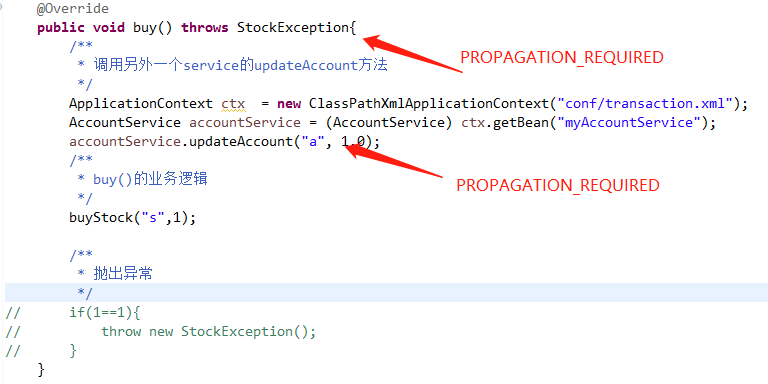
56851 [main] DEBUG org.springframework.jdbc.datasource.DataSourceTransactionManager - Initiating transaction rollback

56852 [main] DEBUG org.springframework.jdbc.datasource.DataSourceTransactionManager - ***Rolling back JDBC transaction on Connection*** [270088606, URL=jdbc:mysql://192.168.168.128:3306/spring, UserName=root@192.168.168.1, MySQL Connector Java]

56856 [main] DEBUG org.springframework.jdbc.datasource.DataSourceTransactionManager - ***Releasing JDBC Connection*** [270088606, URL=jdbc:mysql://192.168.168.128:3306/spring, UserName=root@192.168.168.1, MySQL Connector Java] after transaction

56856 [main] DEBUG org.springframework.jdbc.datasource.DataSourceUtils - Returning JDBC Connection to DataSource

**如果获取需要调用的service的时候，如果重新初始化上下文，是不起作用的，被调用的方法会新启动事务**



555 [main] DEBUG org.springframework.jdbc.datasource.DataSourceTransactionManager - **Creating new transaction with name [cn.pbc.org.cfcc.spring.transaction.StockServiceImpl.buy]**: PROPAGATION\_REQUIRED,ISOLATION\_DEFAULT,-cn.pbc.org.cfcc.spring.transaction.StockException

929 [main] DEBUG org.springframework.jdbc.datasource.DataSourceTransactionManager - Acquired Connection [694079180, URL=jdbc:mysql://192.168.168.128:3306/spring, UserName=root@192.168.168.1, MySQL Connector Java] for JDBC transaction

934 [main] DEBUG org.springframework.jdbc.datasource.DataSourceTransactionManager - Switching JDBC Connection [694079180, URL=jdbc:mysql://192.168.168.128:3306/spring, UserName=root@192.168.168.1, MySQL Connector Java] to manual commit

23620 [main] DEBUG org.springframework.jdbc.datasource.DataSourceTransactionManager - ***Creating new transaction with name [cn.pbc.org.cfcc.spring.transaction.AccountServiceImpl.updateAccount]: PROPAGATION\_REQUIRED,ISOLATION\_DEFAULT***

23715 [main] DEBUG org.springframework.jdbc.datasource.DataSourceTransactionManager - Acquired Connection [1012940229, URL=jdbc:mysql://192.168.168.128:3306/spring, UserName=root@192.168.168.1, MySQL Connector Java] for JDBC transaction

23716 [main] DEBUG org.springframework.jdbc.datasource.DataSourceTransactionManager - Switching JDBC Connection [1012940229, URL=jdbc:mysql://192.168.168.128:3306/spring, UserName=root@192.168.168.1, MySQL Connector Java] to manual commit

23723 [main] DEBUG org.springframework.jdbc.core.JdbcTemplate - Executing prepared SQL update

23726 [main] DEBUG org.springframework.jdbc.core.JdbcTemplate - ***Executing prepared SQL statement [update account set balance=balance-? where aname=?]***

23784 [main] DEBUG org.springframework.jdbc.core.JdbcTemplate - SQL update affected 1 rows

23818 [main] DEBUG org.springframework.jdbc.datasource.DataSourceTransactionManager - Initiating transaction commit

23820 [main] DEBUG org.springframework.jdbc.datasource.DataSourceTransactionManager - ***Committing JDBC transaction on Connection [1012940229, URL=jdbc:mysql://192.168.168.128:3306/spring, UserName=root@192.168.168.1, MySQL Connector Java]***

23828 [main] DEBUG org.springframework.jdbc.datasource.DataSourceTransactionManager - Releasing JDBC Connection [1012940229, URL=jdbc:mysql://192.168.168.128:3306/spring, UserName=root@192.168.168.1, MySQL Connector Java] after transaction

23829 [main] DEBUG org.springframework.jdbc.datasource.DataSourceUtils - Returning JDBC Connection to DataSource

23831 [main] DEBUG org.springframework.jdbc.datasource.DataSourceTransactionManager - ***Resuming suspended transaction after completion of inner transaction***

43791 [main] DEBUG org.springframework.jdbc.core.JdbcTemplate - Executing prepared SQL update

43791 [main] DEBUG org.springframework.jdbc.core.JdbcTemplate - ***Executing prepared SQL statement [update stock set count=count+? where sname=?]***

43794 [main] DEBUG org.springframework.jdbc.core.JdbcTemplate - SQL update affected 1 rows

44856 [main] DEBUG org.springframework.jdbc.datasource.DataSourceTransactionManager - Initiating transaction commit

44857 [main] DEBUG org.springframework.jdbc.datasource.DataSourceTransactionManager - ***Committing JDBC transaction on Connection*** [694079180, URL=jdbc:mysql://192.168.168.128:3306/spring, UserName=root@192.168.168.1, MySQL Connector Java]

44860 [main] DEBUG org.springframework.jdbc.datasource.DataSourceTransactionManager - Releasing JDBC Connection [694079180, URL=jdbc:mysql://192.168.168.128:3306/spring, UserName=root@192.168.168.1, MySQL Connector Java] after transaction

44860 [main] DEBUG org.springframework.jdbc.datasource.DataSourceUtils - Returning JDBC Connection to DataSource

### PROPAGATION\_SUPPORTS

如果当前在事务中，即以事务的形式运行，如果当前不在一个事务中，那么就以非事务的形式运行

### PROPAGATION\_MANDATORY

必须在一个已有的事务中执行，否则抛出异常



org.springframework.transaction.IllegalTransactionStateException: No existing transaction found for transaction marked with propagation 'mandatory'

at org.springframework.transaction.support.AbstractPlatformTransactionManager.getTransaction(AbstractPlatformTransactionManager.java:359)

at org.springframework.transaction.interceptor.TransactionAspectSupport.createTransactionIfNecessary(TransactionAspectSupport.java:447)

at org.springframework.transaction.interceptor.TransactionAspectSupport.invokeWithinTransaction(TransactionAspectSupport.java:277)

at org.springframework.transaction.interceptor.TransactionInterceptor.invoke(TransactionInterceptor.java:96)

at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:179)

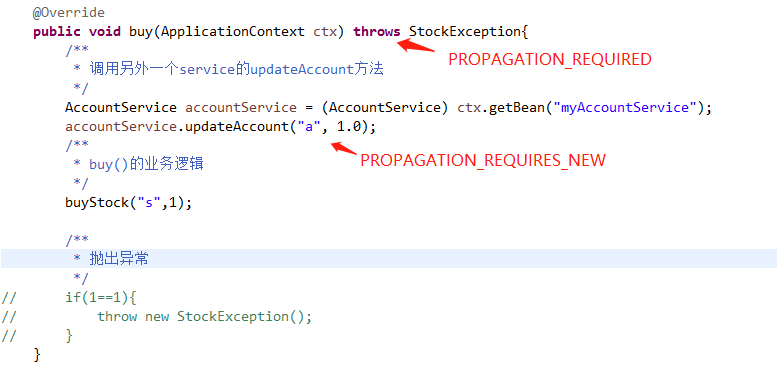
at org.springframework.aop.framework.JdkDynamicAopProxy.invoke(JdkDynamicAopProxy.java:213)

at com.sun.proxy.$Proxy8.updateAccount(Unknown Source)

at cn.pbc.org.cfcc.spring.transaction.StockServiceImpl.buy(StockServiceImpl.java:27)

### PROPAGATION\_REQUIRES\_NEW

不管是否存在事务，都创建一个新的事务，原来的挂起，新的执行完毕，继续执行老的事务



478 [main] DEBUG org.springframework.jdbc.datasource.DataSourceTransactionManager - ***Creating new transaction with name [cn.pbc.org.cfcc.spring.transaction.StockServiceImpl.buy]:*** PROPAGATION\_REQUIRED,ISOLATION\_DEFAULT,-cn.pbc.org.cfcc.spring.transaction.StockException

792 [main] DEBUG org.springframework.jdbc.datasource.DataSourceTransactionManager - Acquired Connection [1273284163, URL=jdbc:mysql://192.168.168.128:3306/spring, UserName=root@192.168.168.1, MySQL Connector Java] for JDBC transaction

797 [main] DEBUG org.springframework.jdbc.datasource.DataSourceTransactionManager - Switching JDBC Connection [1273284163, URL=jdbc:mysql://192.168.168.128:3306/spring, UserName=root@192.168.168.1, MySQL Connector Java] to manual commit

798 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory - Returning cached instance of singleton bean 'myAccountService'

798 [main] DEBUG org.springframework.jdbc.datasource.DataSourceTransactionManager - ***Suspending current transaction, creating new transaction with name [cn.pbc.org.cfcc.spring.transaction.AccountServiceImpl.updateAccount]***

812 [main] DEBUG org.springframework.jdbc.datasource.DataSourceTransactionManager - Acquired Connection [1773684356, URL=jdbc:mysql://192.168.168.128:3306/spring, UserName=root@192.168.168.1, MySQL Connector Java] for JDBC transaction

812 [main] DEBUG org.springframework.jdbc.datasource.DataSourceTransactionManager - Switching JDBC Connection [1773684356, URL=jdbc:mysql://192.168.168.128:3306/spring, UserName=root@192.168.168.1, MySQL Connector Java] to manual commit

814 [main] DEBUG org.springframework.jdbc.core.JdbcTemplate - Executing prepared SQL update

815 [main] DEBUG org.springframework.jdbc.core.JdbcTemplate - ***Executing prepared SQL statement [update account set balance=balance-? where aname=?]***

830 [main] DEBUG org.springframework.jdbc.core.JdbcTemplate - SQL update affected 1 rows

842 [main] DEBUG org.springframework.jdbc.datasource.DataSourceTransactionManager - Initiating transaction commit

842 [main] DEBUG org.springframework.jdbc.datasource.DataSourceTransactionManager - C***ommitting JDBC transaction on Connection*** [1773684356, URL=jdbc:mysql://192.168.168.128:3306/spring, UserName=root@192.168.168.1, MySQL Connector Java]

846 [main] DEBUG org.springframework.jdbc.datasource.DataSourceTransactionManager - ***Releasing JDBC Connection*** [1773684356, URL=jdbc:mysql://192.168.168.128:3306/spring, UserName=root@192.168.168.1, MySQL Connector Java] after transaction

846 [main] DEBUG org.springframework.jdbc.datasource.DataSourceUtils - Returning JDBC Connection to DataSource

847 [main] DEBUG org.springframework.jdbc.datasource.DataSourceTransactionManager - ***Resuming suspended transaction after completion of inner transaction***

847 [main] DEBUG org.springframework.jdbc.core.JdbcTemplate - Executing prepared SQL update

847 [main] DEBUG org.springframework.jdbc.core.JdbcTemplate - Executing prepared SQL statement [update stock set count=count+? where sname=?]

849 [main] DEBUG org.springframework.jdbc.core.JdbcTemplate - SQL update affected 1 rows

849 [main] DEBUG org.springframework.jdbc.datasource.DataSourceTransactionManager - Initiating transaction commit

850 [main] DEBUG org.springframework.jdbc.datasource.DataSourceTransactionManager - ***Committing JDBC transaction on Connection*** [1273284163, URL=jdbc:mysql://192.168.168.128:3306/spring, UserName=root@192.168.168.1, MySQL Connector Java]

852 [main] DEBUG org.springframework.jdbc.datasource.DataSourceTransactionManager - ***Releasing JDBC Connection*** [1273284163, URL=jdbc:mysql://192.168.168.128:3306/spring, UserName=root@192.168.168.1, MySQL Connector Java] after transaction

852 [main] DEBUG org.springframework.jdbc.datasource.DataSourceUtils - Returning JDBC Connection to DataSource

### PROPAGATION\_NOT\_SUPPORTED

这个方法不开启事务

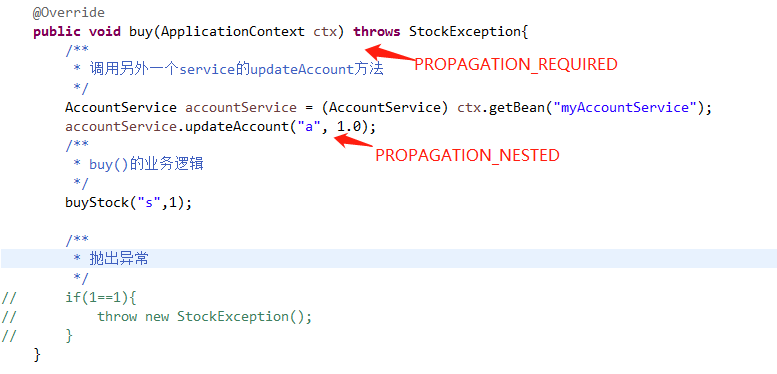
### PROPAGATION\_NEVER

不能在一个事务中执行，就是当前必须没有事务，否则抛出异常。ServiceA.methodA调用ServiceB.methodB，那么当执行到ServiceB.methodB时，ServiceA.methodA的事务挂起，而它以非事务的状态运行完，再继续ServiceA.methodA的事务。

### PROPAGATION\_NESTED

如果一个活动的事务存在，则运行在一个嵌套的事务中，如果没有活动的事务，则按照REQUIRED属性执行，它使用一个单独的事务。这个事务拥有多个回滚的保存点，内部事务的回滚不会对外部事务造成影响，它只对DataSource TransactionManager事务管理器起效。

理解Nested的关键是savepoint。他与PROPAGATION\_REQUIRES\_NEW的区别是，PROPAGATION\_REQUIRES\_NEW另起一个事务，将会与他的父事务相互独立，而Nested的事务和他的父事务是相依的，他的提交是要等和他的父事务一块提交的。也就是说，如果父事务最后回滚，他也要回滚的。而Nested事务的好处是他有一个savepoint。



505 [main] DEBUG org.springframework.jdbc.datasource.DataSourceTransactionManager - ***Creating new transaction with name [cn.pbc.org.cfcc.spring.transaction.StockServiceImpl.buy]:*** PROPAGATION\_REQUIRED,ISOLATION\_DEFAULT,-cn.pbc.org.cfcc.spring.transaction.StockException

827 [main] DEBUG org.springframework.jdbc.datasource.DataSourceTransactionManager - Acquired Connection [1207598037, URL=jdbc:mysql://192.168.168.128:3306/spring, UserName=root@192.168.168.1, MySQL Connector Java] for JDBC transaction

831 [main] DEBUG org.springframework.jdbc.datasource.DataSourceTransactionManager - Switching JDBC Connection [1207598037, URL=jdbc:mysql://192.168.168.128:3306/spring, UserName=root@192.168.168.1, MySQL Connector Java] to manual commit

832 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory - Returning cached instance of singleton bean 'myAccountService'

832 [main] DEBUG org.springframework.jdbc.datasource.DataSourceTransactionManager - ***Creating nested transaction with name [cn.pbc.org.cfcc.spring.transaction.AccountServiceImpl.updateAccount]***

842 [main] DEBUG org.springframework.jdbc.core.JdbcTemplate - Executing prepared SQL update

842 [main] DEBUG org.springframework.jdbc.core.JdbcTemplate - Executing prepared SQL statement [update account set balance=balance-? where aname=?]

857 [main] DEBUG org.springframework.jdbc.core.JdbcTemplate - SQL update affected 1 rows

867 [main] ***DEBUG org.springframework.jdbc.datasource.DataSourceTransactionManager - Releasing transaction savepoint***

867 [main] DEBUG org.springframework.jdbc.core.JdbcTemplate - Executing prepared SQL update

867 [main] DEBUG org.springframework.jdbc.core.JdbcTemplate - Executing prepared SQL statement [update stock set count=count+? where sname=?]

868 [main] DEBUG org.springframework.jdbc.core.JdbcTemplate - SQL update affected 1 rows

869 [main] DEBUG org.springframework.jdbc.datasource.DataSourceTransactionManager - Initiating transaction commit

869 [main] DEBUG org.springframework.jdbc.datasource.DataSourceTransactionManager - Committing JDBC transaction on Connection [1207598037, URL=jdbc:mysql://192.168.168.128:3306/spring, UserName=root@192.168.168.1, MySQL Connector Java]

873 [main] DEBUG org.springframework.jdbc.datasource.DataSourceTransactionManager - Releasing JDBC Connection [1207598037, URL=jdbc:mysql://192.168.168.128:3306/spring, UserName=root@192.168.168.1, MySQL Connector Java] after transaction

873 [main] DEBUG org.springframework.jdbc.datasource.DataSourceUtils - Returning JDBC Connection to DataSource

## Spring分布式事务

<https://blog.csdn.net/liaohaojian/article/details/68488150>

# CGLIB

代理类将委托类作为自己的父类并为其中的非final委托方法创建两个方法，一个是与委托方法签名相同的方法，它在方法中会通过super调用委托方法；另一个是代理类独有的方法。在代理方法中，它会判断是否存在实现了MethodInterceptor接口的对象，若存在则将调用intercept方法对委托方法进行代理

可以在运行时对类或者是接口进行增强操作，且委托类无需实现接口。

不能对final类以及final方法进行代理

底层将方法全部存入一个数组中，通过数组索引直接进行方法调用