一、spring

1、Spring简介

1.1 spring的优势

（1）方便解耦，简化开发

（2）AOP编程的支持

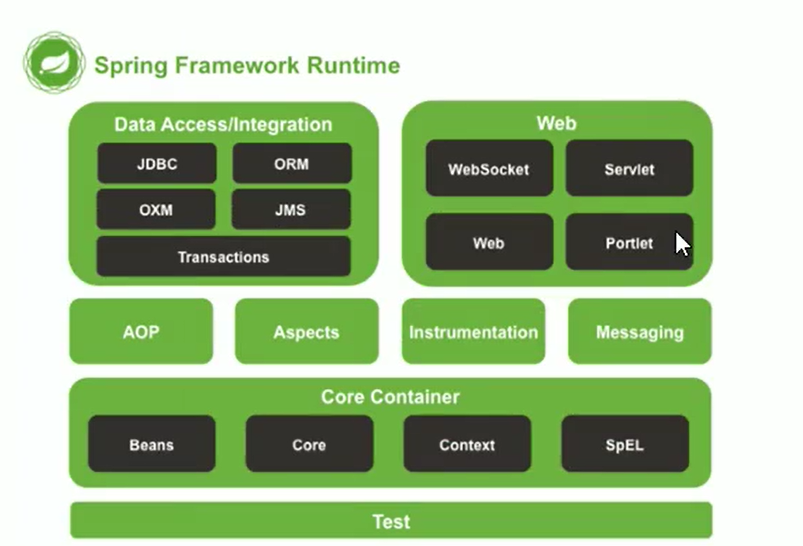
（3）声明式事务的支持

（4）方便集成各种优秀的框架

spring对各种优秀的框架(Struts, Hibernate, Hessian, Quartz)的支持

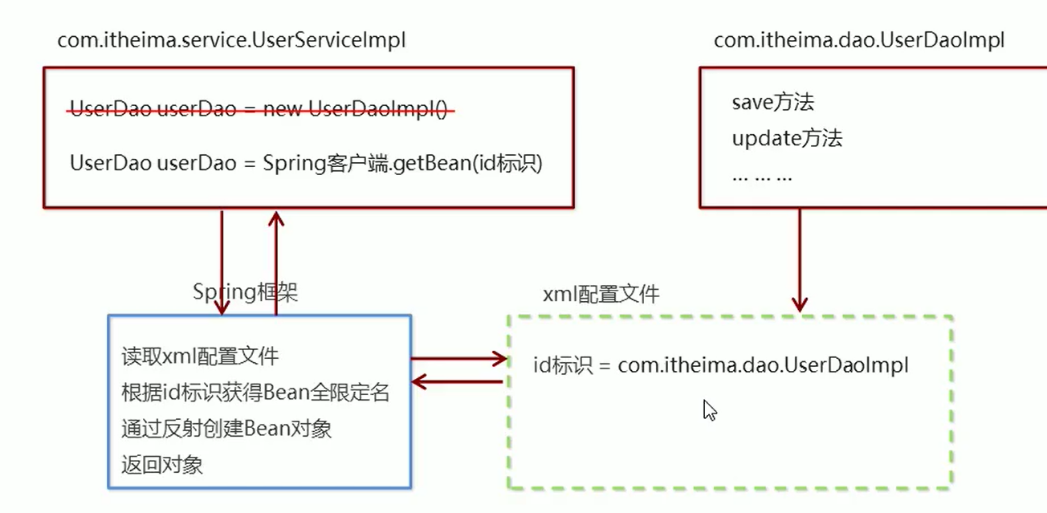
（5）降低JavaEE API的使用难度

1.2 spring的体系结构



2、Spring的快速入门

2.1 程序的开发步骤



1. 导入Spring开发的基本包坐标
2. 编写Dao接口和实现类
3. 创建Spring核心配置文件
4. 在Spring配置文件中配置UserDaoImpl
5. 使用Spring的API获得Bean实例

3、Spring配置文件

3.1 Bean标签基本配置

用于配置对象交由spring创建，默认情况下它调用的是类中的无参数构造函数，如果没有无参数构造函数则不能创建成功。

基本属性：

id: Bean实例在Spring容器中的唯一标识

class：Bean的全限定名称

3.2 Bean标签的范围配置

scope: 指对象的作用范围，取值如下：



1. 当scope的取值是singleton时：

Bean的实例化个数： 1个

Bean的实例化时机：当Spring核心文件被加载时，实例化配置的Bean实例

Bean的生命周期：

对象创建：当应用加载，创建容器时，对象就被创建了

对象运行：只要容器在，对象一直活着

对象销毁：当应用卸载，销毁容器时，对象就被销毁了

1. 当scope的取值是prototype时：

Bean的实例化个数： 多个

Bean的实例化时机：当调用getBean()方法时实例化Bean

Bean的生命周期：

对象创建：当使用对象时，创建新的对象

对象运行：只要对象在使用中，就一直活着

对象销毁：当对象长时间不使用时，被java垃圾回收器回收

3.3 Bean生命周期配置

init-method：指定类中的初始化方法名称

destroy-method：指定类中销毁方法名称

3.4 Bean的实例化三种方式

无参构造方法实例化

工厂静态方法实例化

工厂实例方法实例化

3.5 Bean的依赖注入

依赖注入(Dependency Injection): 它是Spring框架核心IOC的具体实现

注入方式：

1. set方法注入

例：<property name="userDao" ref="userDao"/>

P命名空间注入本质也是set方法注入，但比起上述的set方法注入更加方便，主要体现在配置文件中，如下：

首先，需要引入命名空间：

xmlns:p=”<http://www.springframework.org/schema/p>”

其次，需要修改注入方式

<bean id="userService" class="com.xian.service.impl.UserServiceImpl" p:userDao-ref="userDao"/>

构造方法注入

3.6 Bean的依赖注入的数据类型

除了对象的引用可以注入，普通数据类型，集合等都可以在容器中进行注入

注入数据的三种数据类型：

普通数据类型

引用数据类型

集合数据类型

3.7 引入其他的配置文件

在spring主配置文件中通过import标签进行加载

<import resource=”applicationContext-xxx.xml”/>

Spring的重点配置：

<bean>标签

id属性: 在容器中Bean实例的唯一标识，不允许重复

class属性：要实例化的Bean的全限定名

scope属性：Bean的作用范围，常用的是singleton和prototype

<property> 标签: 属性注入

name属性： 属性名称

value属性：注入的普通属性值

ref属性：注入的对象引用值

<list>标签

<map>标签

<properties>标签

<contructor-arg>标签

<import>标签：导入其他的spring的分文件

4、Spring相关API

4.1 ApplicationContext的继承体系

applicationContext: 接口类型，代表应用上下文，可以通过其实例获得Spring容器中的Bean对象

4.2 ApplicationContext的实现类

(1) ClassPathXmlApplicationContext

它是从类的根路径下加载配置文件，推荐使用

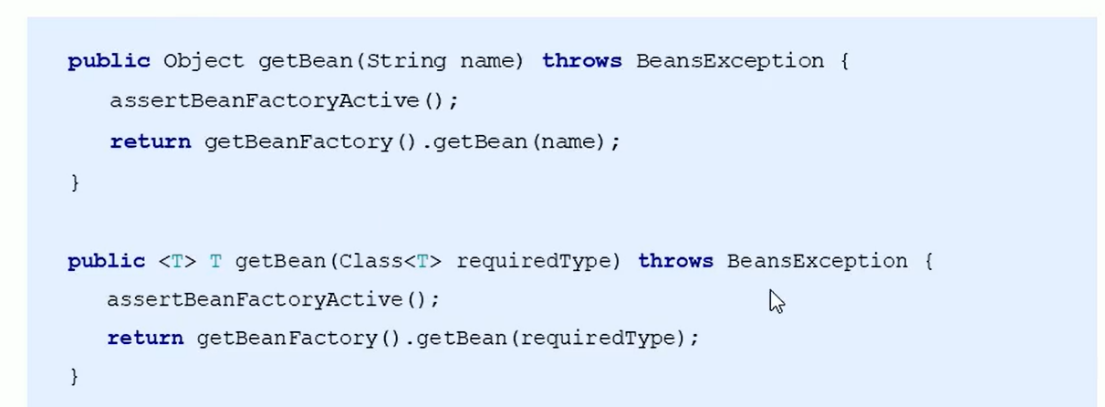
(2) FileSystemXmlApplicationContext

它是从磁盘路径上加载配置文件

(3) AnnotationConfigApplicationContext

当使用注解配置容器对象时，需要使用此类来创建spring容器。它用来读取注解

4.3 getBean()方法的使用



4.4 知识要点

Spring的重点API



5、Spring配置数据源

5.1、数据源(连接池)的作用

数据源是提高程序性能出现的

事先实例化数据源，初始化部分连接资源

使用连接资源时从数据源中获取

使用完毕后将连接资源归还给数据源

常见的数据源：DBCP、C3p0、BoneCP、Druid等

5.2 数据源的创建，见代码

5.3 Spring产生数据源对象，见代码

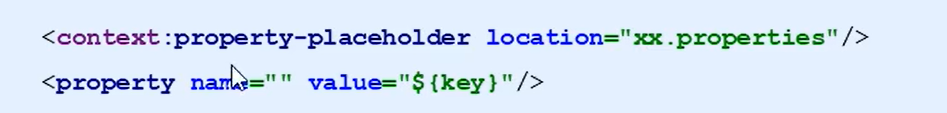
5.4 抽取jdbc配置文件

applicationContext.xml加载jdbc.properties配置文件获得连接信息



5.5 知识要点

spring加载properties文件



6、Spring注解开发

6.1、spring原始注解

spring是轻代码而重配置的框架，配置比较繁重，影响开发效率，所以注解开发是一种趋势，注解代替xml配置文件可以简化配置，提高开发效率。

spring原始注解主要是替代Bean



注意：

使用注解进行开发时，需要在applicationContext.xml中配置组件扫描，作用是指定哪个包及包下的 Bean需要进行扫描以便识别使用注解配置的类、字段和方法。



6.2、spring新注解



7、Spring整合Junit

7.1、原始Junit测试Spring的问题

在测试类中，每个测试方法都有以下两行代码：



这两行代码的作用是获取容器，如果不写，直接会提示空指针异常。

7.2、上述问题解决思路

让SpringJunit负责创建Spring容器，但是需要将配置文件的名称告诉它；

将需要进行测试的Bean直接在测试类中进行注入；

7.3、Spring集成Junit步骤

① 导入spring集成Junit坐标

② 使用@Runwith注解替换原来的运行期

③ 使用@ContextConfiguration指定配置文件或配置类

④ 使用@Autowired注入需要测试的对象

⑤ 创建测试方法进行测试

8、Spring的AOP简介

8.1、什么是AOP

Aspect Oriented Programming，面向切面编程，通过预编译方式和运行期动态代理实现程序功能的统一维护的一种技术。

8.2、AOP的作用及其优势

作用：在程序运行期间，在不修改源码的情况下对方法进行功能增强

优势：减少重复代码，提高开发效率，并且便于维护

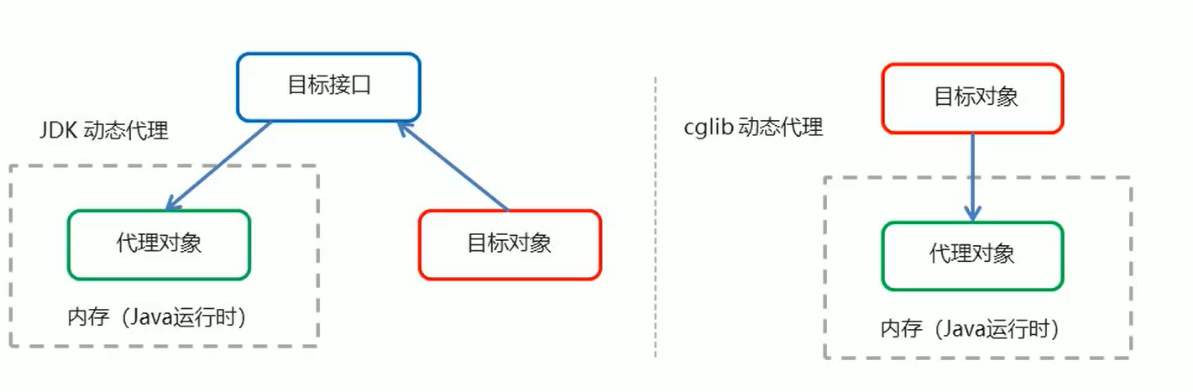
8.3、AOP的底层实现

实际上，AOP的底层是通过Spring提供的动态代理技术实现的。在运行期间，Spring通过动态代理技术动态的生成代理对象，代理对象方法执行时进行增强功能的介入，在去调用目标对象得到方法，从而完成功能的增强。

8.4、AOP的动态代理技术

JDK代理：基于接口的动态代理技术

cglib代理：基于父类的动态代理技术



8.5、AOP的相关概念

Target(目标对象)：代理的目标对象

Proxy(代理对象)：一个代理类

JointPoint(连接点)：指那些被拦截到的点。在spring中指的是方法，即可以被增强的方法。因为spring只支持方法类型的连接点。

PointCut(切入点)：指我们要对哪些JointPoint进行拦截的定义。

Advice(通知/增强)：所谓通知指拦截到JointPoint之后要做的事情就是通知。

Aspect(切面)：切入点和通知的结合

Weaving(织入)：指把增强方法应用到目标对象来创建新的代理对象的过程。Spring采用动态代理织入， 而AspectJ采用编译期织入和类装载期织入

8.6、AOP开发明确的事项

8.6.1、需要编写的内容

业务核心代码(目标类的目标方法)

切面类，切面类中有通知

在配置文件中，配置织入关系，即将哪些通知与哪些连接点进行结合

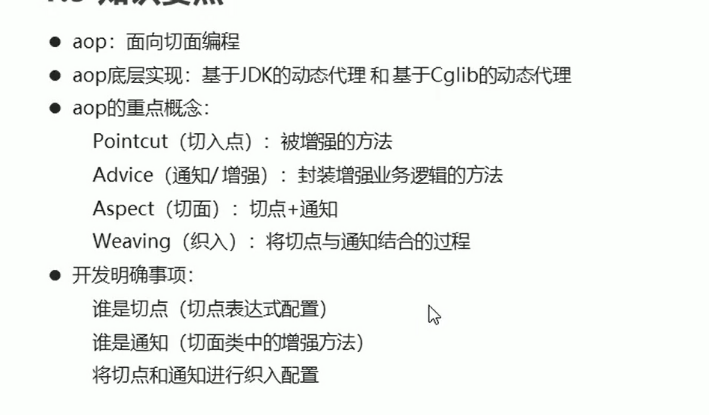
8.6.2、AOP技术实现的内容

Spring框架监控切入点方法的执行。一旦监控到切入点方法被运行，使用代理机制，动态创建目标对象的代理对象，根据通知类别，在代理对象的对应位置，将通知对应的功能织入，完成完整的代码逻辑运行。

8.6.3、AOP底层使用哪种代理方式

在spring中，框架会根据目标类是否实现了接口来决定采用哪种动态代理的方式。

8.7、知识要点



9、基于XML的AOP开发

9.1、快速入门

① 导入AOP相关坐标

② 创建目标接口和目标类(内部有切入点)

③ 创建切面类(内部有增强方法)

④ 将目标类和切面类的对象创建交给spring

⑤ 在applicationContext.xml中配置织入关系

⑥ 测试代码

9.2 XML配置AOP详解

(1) 切点表达式语法

execution([修饰符] 返回值类型 包名.类名.方法名(参数))

·访问修饰符可以省略

·返回值类型、包名、类名、方法名可以使用星号\*代表任意

·包名与类名之间一个点 . 代表当前包下的类，两个点 .. 表示当前包及其子包下的类

·参数列表可以使用两个 .. 表示任意个数，任意类型的参数列表

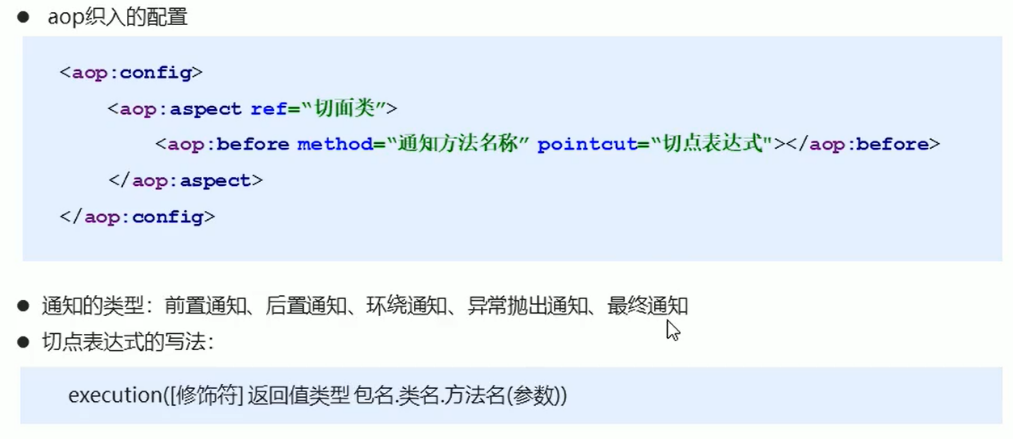
(2) 通知的类型



(3) 切点表达式的抽取



9.3、知识要点



10、基于注解的AOP开发

10.1、快速入门

① 创建目标接口和目标类(内部有切入点)

② 创建切面类(内部有增强方法)

③ 将目标类和切面类的对象创建交给spring

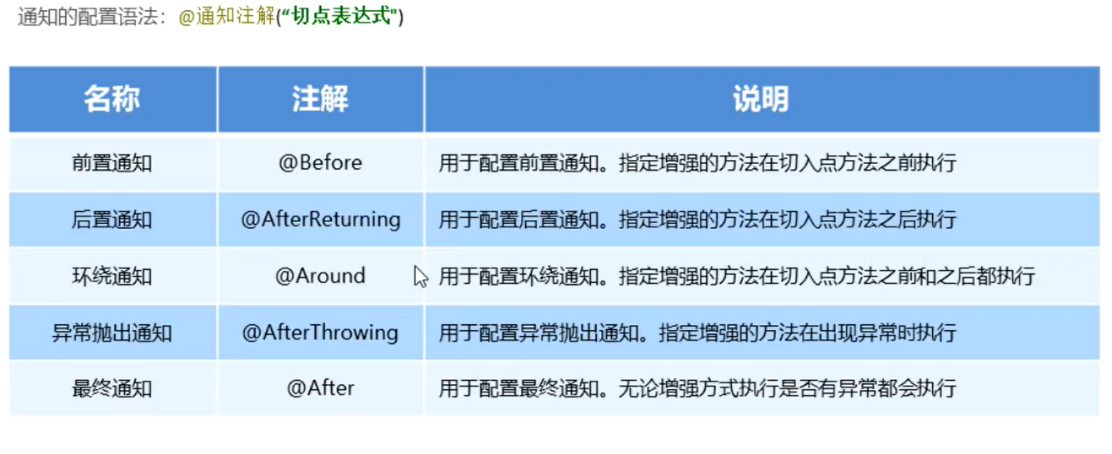
④ 在切面类中使用注解配置织入关系

⑤ 在配置文件中开启组件扫描和AOP的自动代理

⑥ 测试代码

10.2、注解配置AOP详解

(1)、注解通知的类型



(2)、切点表达式的抽取



11、Spring JdbcTemplate基本使用

11.1、JdbcTemplate开发步骤

① 导入spring-jdbc和spring-tx坐标

② 创建数据库表和实体

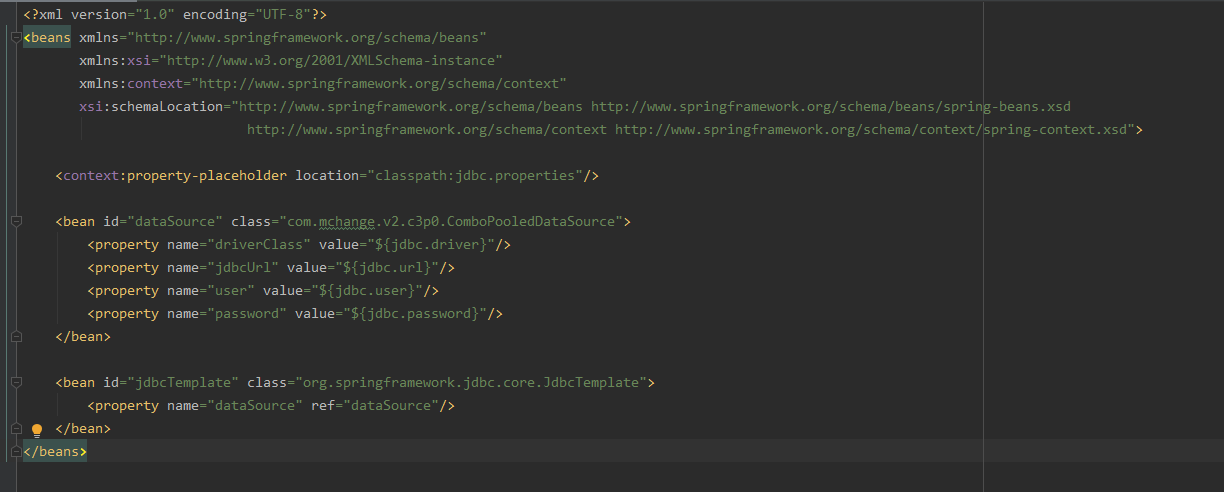
③ 创建JdbcTemplate对象

④ 执行数据库操作

11.2、Spring产生JdbcTemplate对象



用context提取配置文件



11.3、CURD 见代码

11.4、知识要点

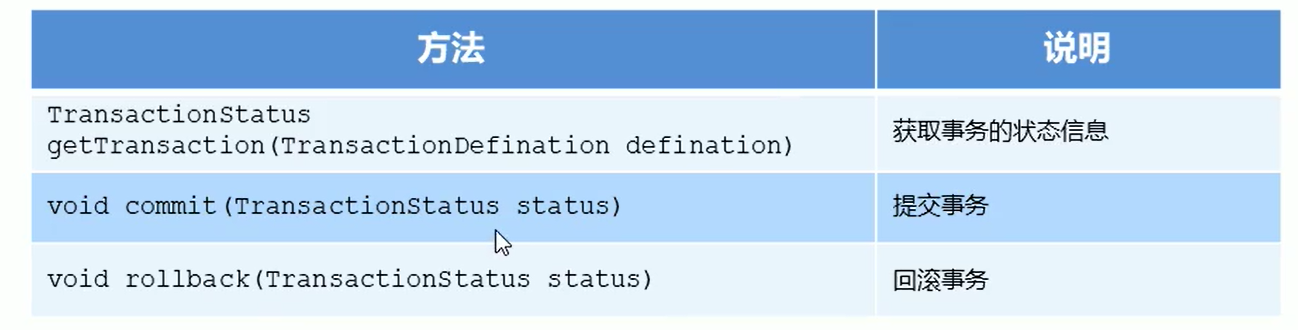


12、Spring的事务控制

12.1、编程式事务控制相关对象

1、PlantformTransactionManager

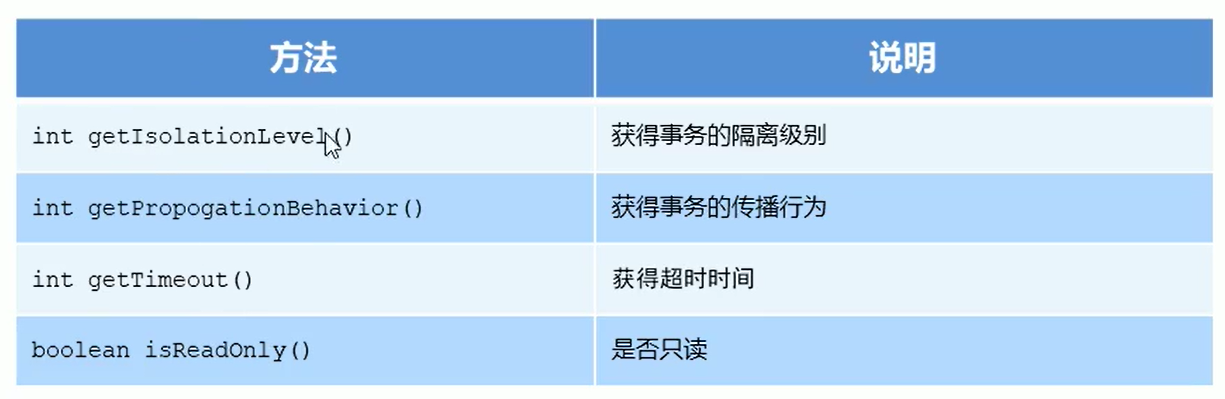
PlantformTransactionManager接口是spring的事务管理器，它里面提供了我们常用的操作事务的方法。





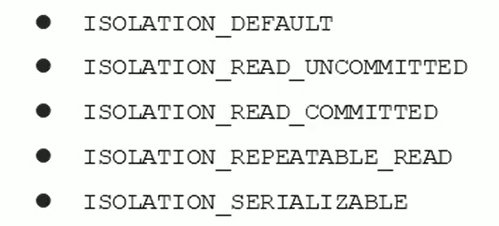
2、TransactionDefinition

TransactionDefinition是事务的定义信息对象，里面有如下方法：

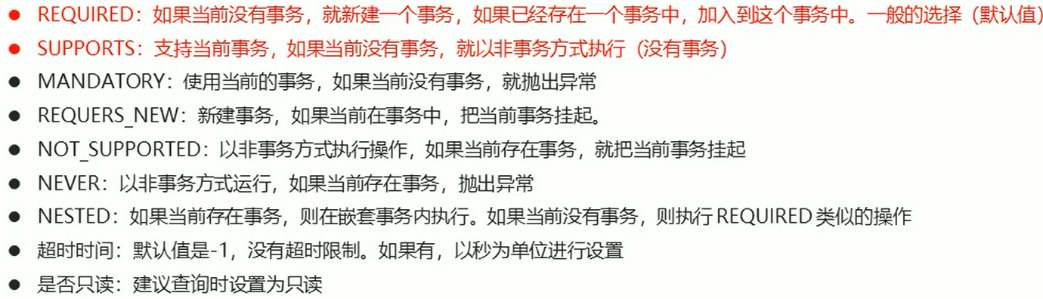


(1)事务的隔离级别

设置事务的隔离级别，可以解决事务并发产生的问题，如脏读、不可重复读、虚读。

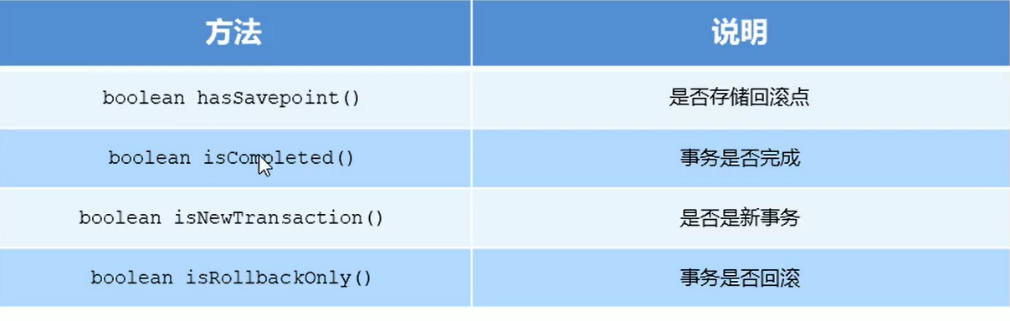


(2)事务的传播行为



3、TransactionStatus

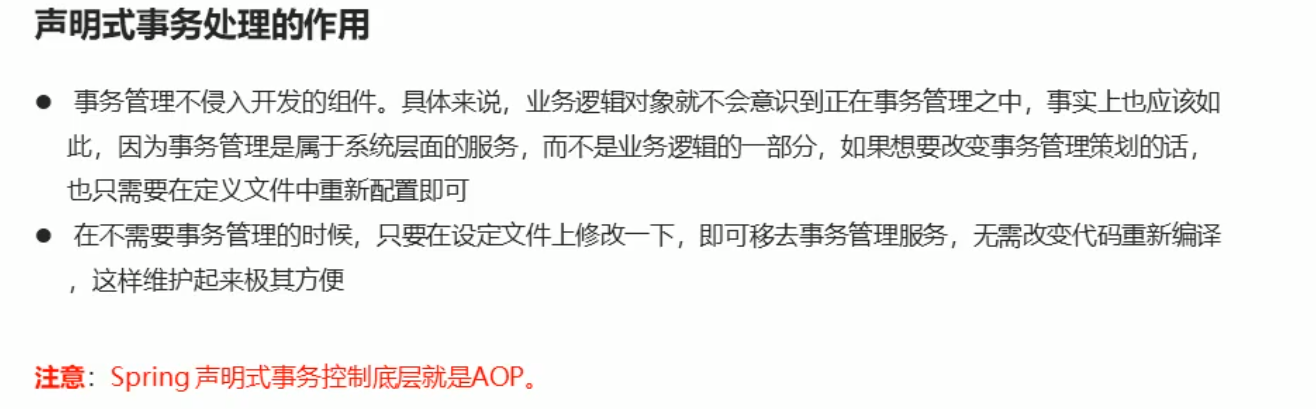
TransactionStatus接口提供的是事务具体运行状态，方法介绍如下



12.2、基于XML的声明式事务控制

12.2.1、什么是声明式事务控制

spring的声明式事务顾名思义就是采用声明的方式开来处理事务。这里所说的声明，就是指在配置文件中声明，用在spring配置文件中声明式的处理事务来代替代码式的处理事务。



12.2.2、声明式事务控制的实现

声明式事务控制明确事项：

谁是切点？

谁是通知？

配置切面？

12.2.3、切点方法的事务参数的配置



12.2.4、知识要点

声明式事务配置要点：

* 平台事务管理器配置
* 事务通知的配置
* 事务aop织入的配置

12.3、基于注解的声明式事务控制

见代码

