Spring 基础搭建 以及 bean值的注入

--------------------------------------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------------------------------------

EE 开发分成三层结构:

\* WEB 层:Spring MVC.

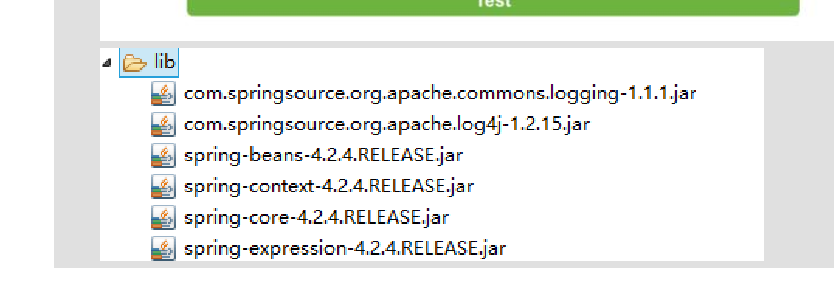
\* 业务层:Bean 管理:(IOC)

\* 持久层:Spring 的 JDBC 模板.ORM 模板用于整合其他的持久层框架.

基本spring搭建

1. 导包





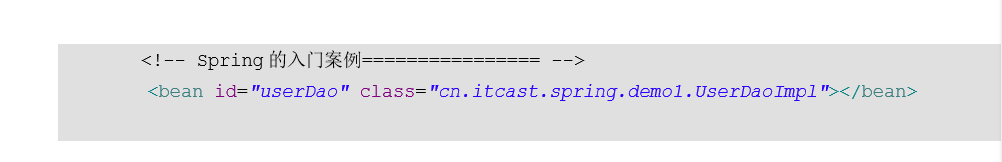
1. 配置基本的XML等等



1. 类与接口



1. 完成类的相关XML



1. 测试

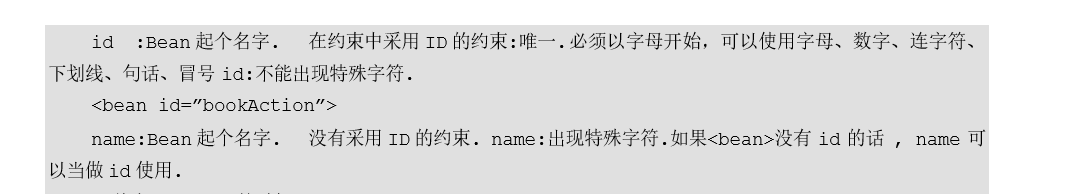


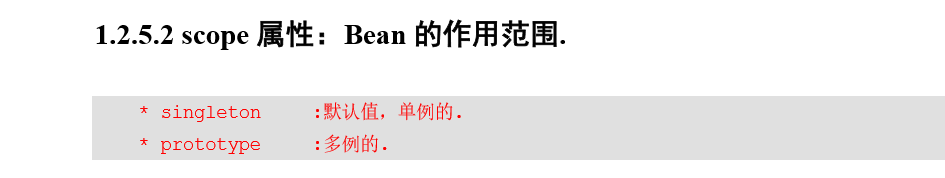
IOC :控制反转,将对象的创建权交给了 Spring.

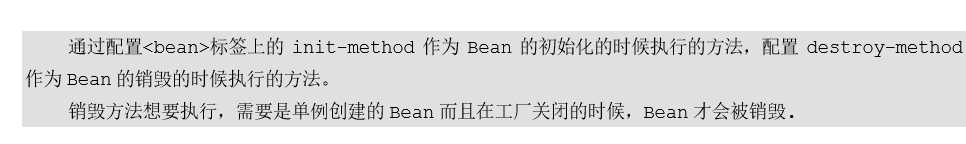
DI :Dependency Injection 依赖注入.需要有IOC的环境,Spring创建这个类的过程中,Spring将类的依 赖的属性设置进去.

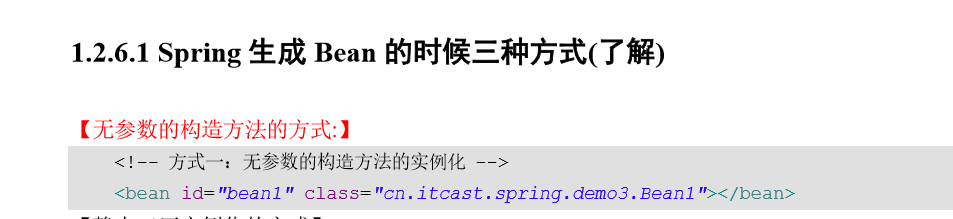
BeanFactory :是在 getBean 的时候才会生成类的实例. ApplicationContext :在加载 applicationContext.xml(容器启动)时候就会创建.

XML的提示配置

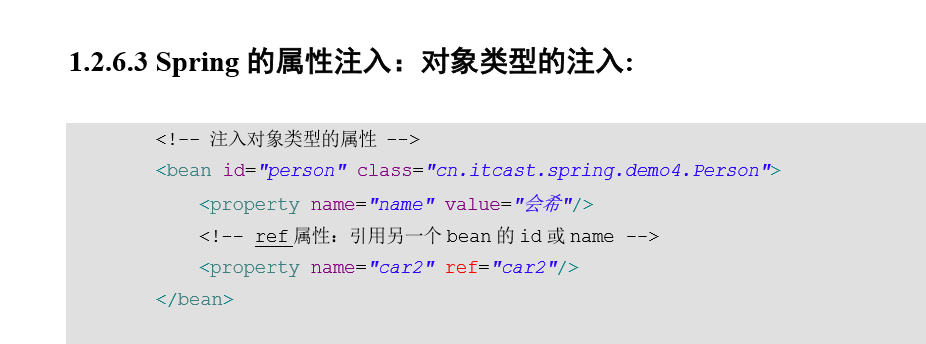








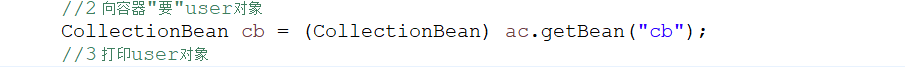


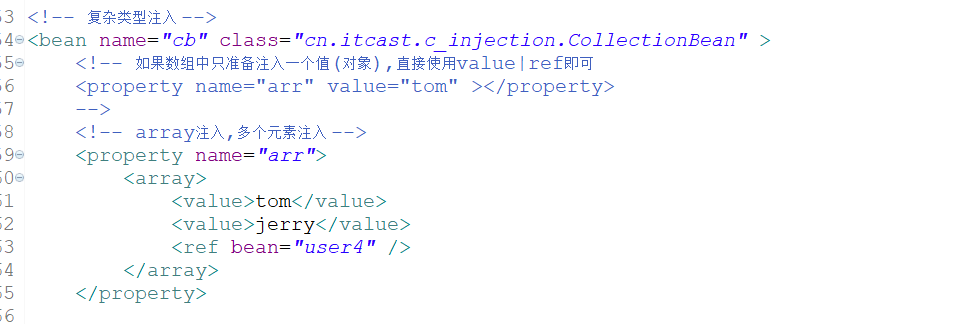






总结：







--------------------------------------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------------------------------------

Spring 常规注解以及 基于 AspectJ 的 AOP 开发

--------------------------------------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------------------------------------

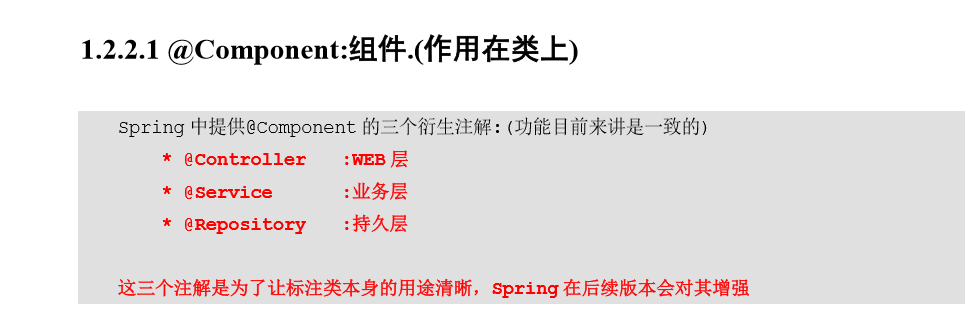
在基础之上

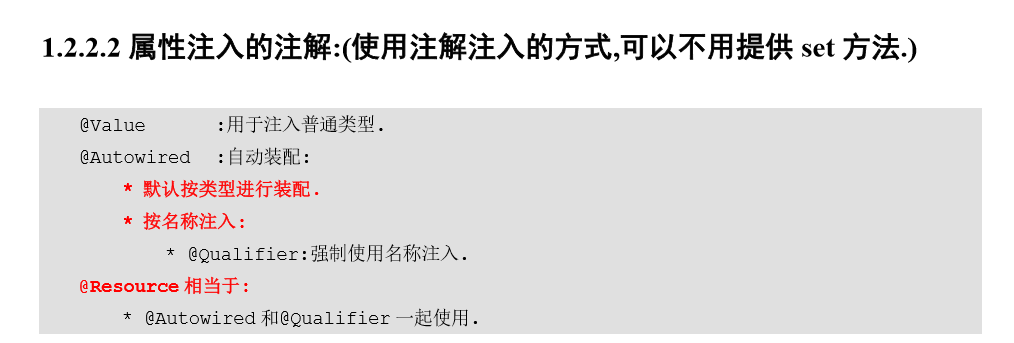




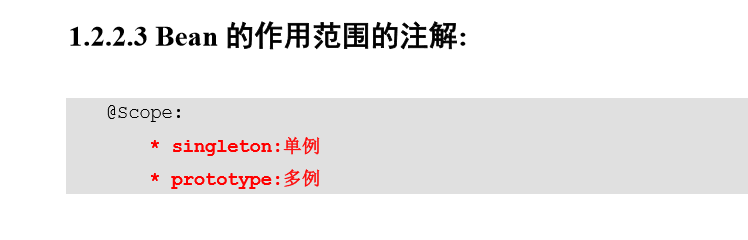


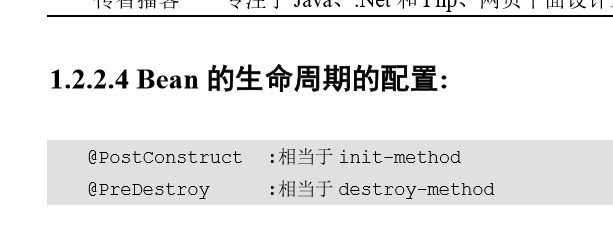
就是加了一个注解其他没有变



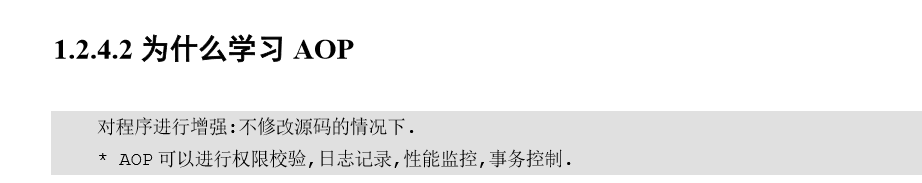


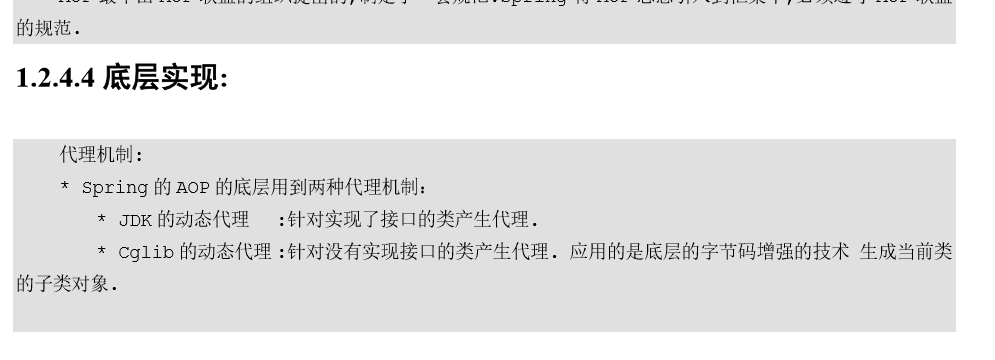
Set方法不就是之前 bean下面的值注入

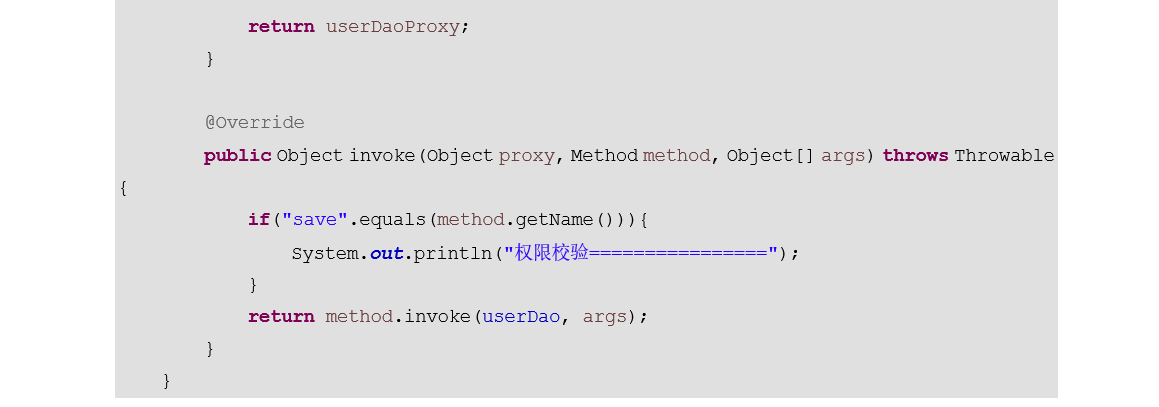


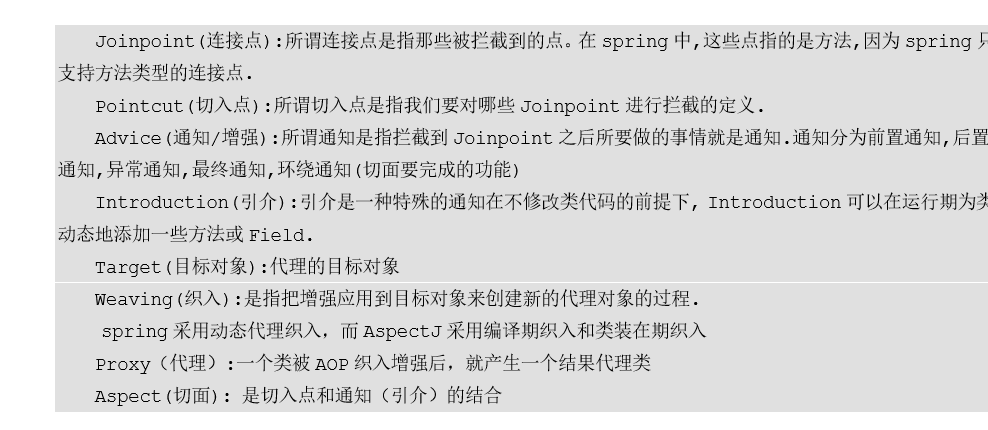








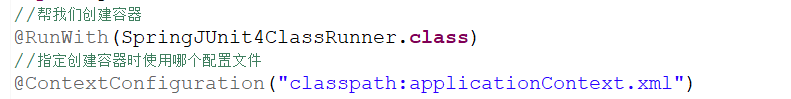








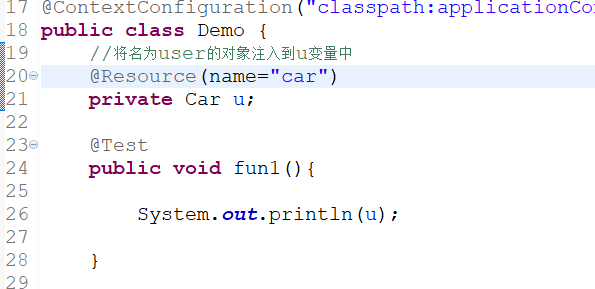
注解的实现：

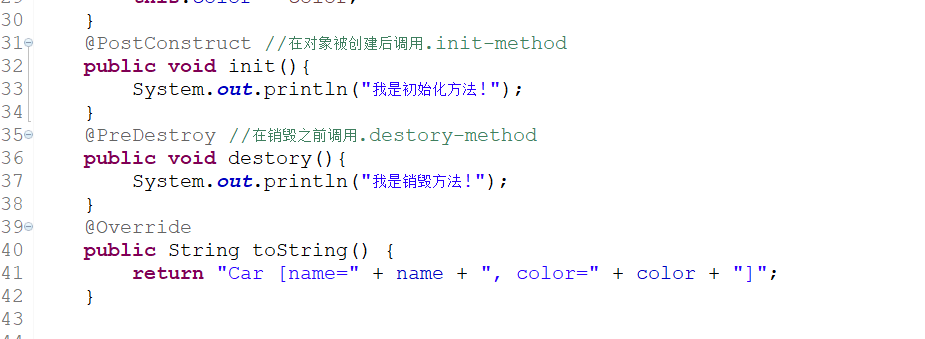
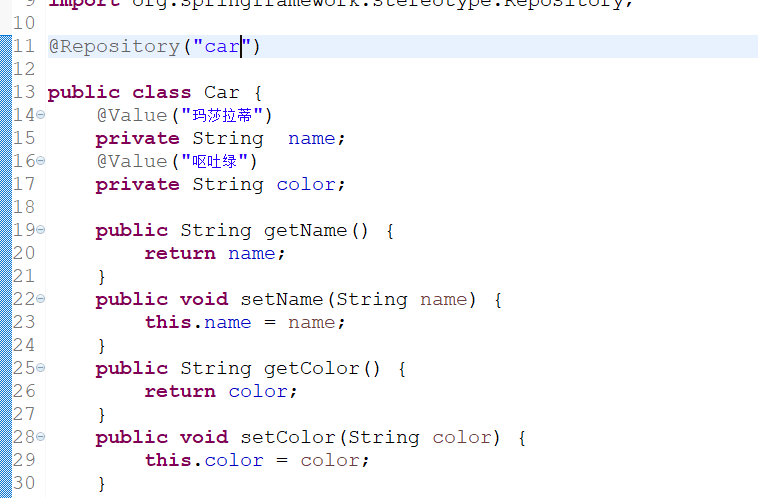


这种写法是为了让测试在Spring容器环境下执行。

Spring的容器环境是啥呢？

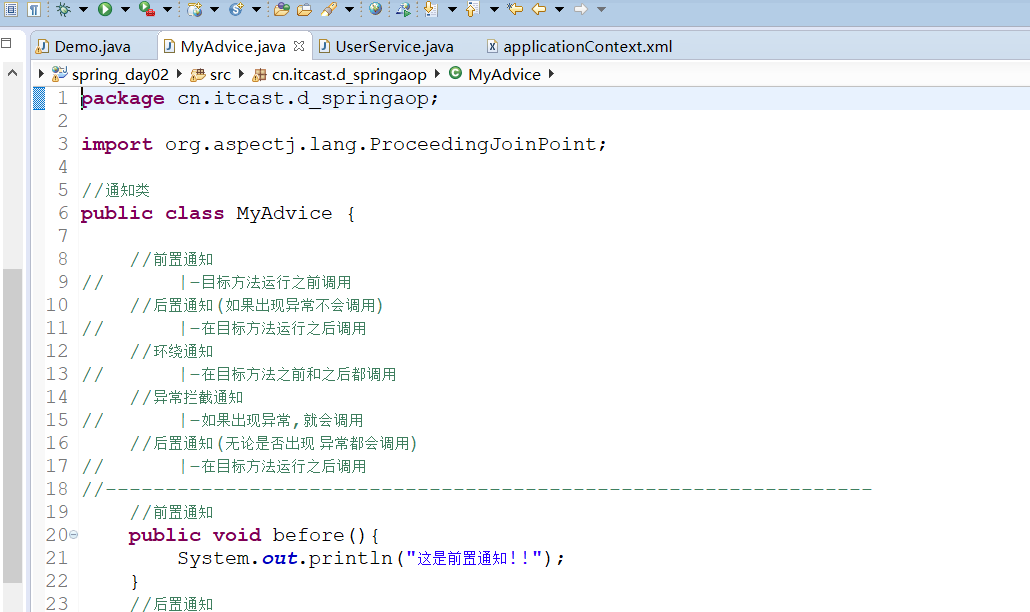
比如常见的 Service  Dao  Action ， 这些个东西，都在Spring容器里，junit需要将他们拿到，并且使用来测试。

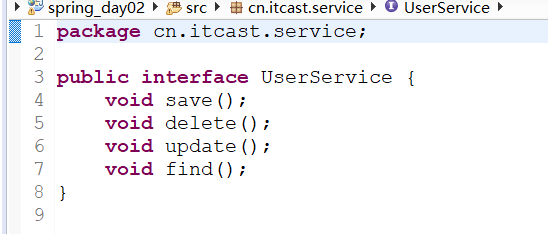
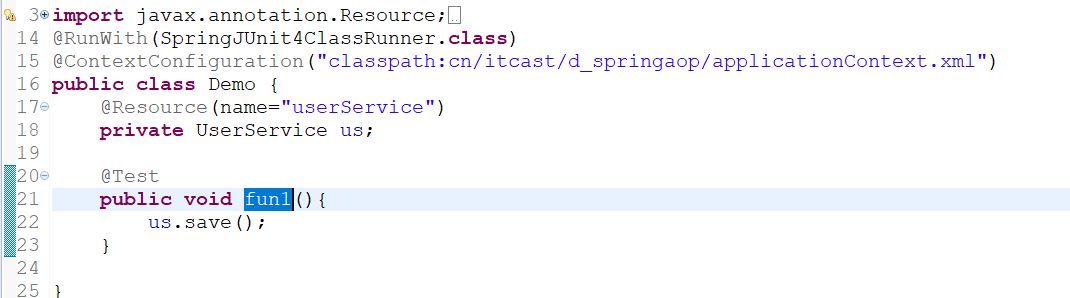




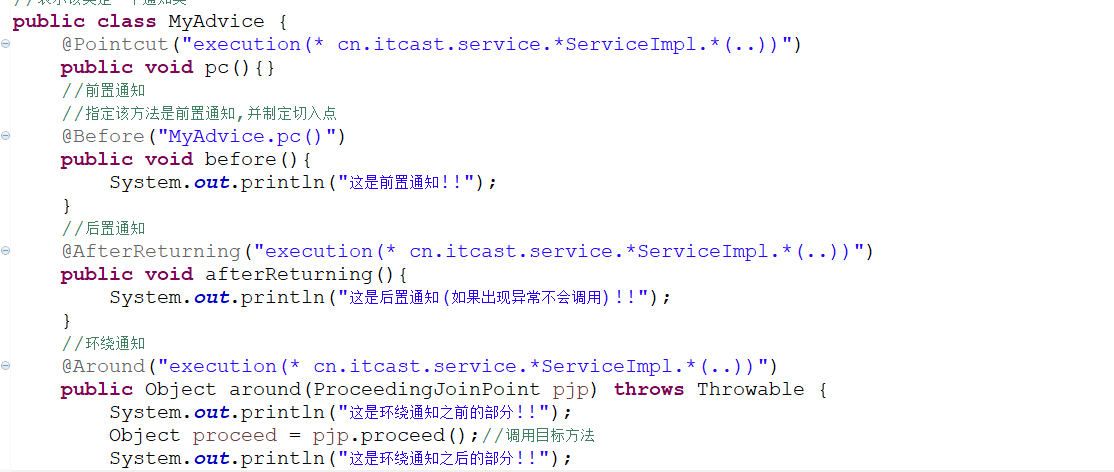
AOP切面增强方法：

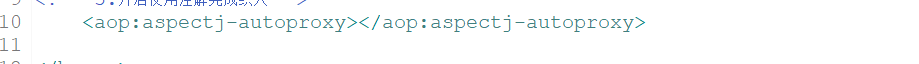






当然也可以使用注解来





--------------------------------------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------------------------------------

Spring JDBC Xxxx 事务

--------------------------------------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------------------------------------

1.4.4 事务的回顾:

1.4.4.1 什么是事务:

事务逻辑上的一组操作,组成这组操作的各个逻辑单元,要么一起成功,要么一起失败.

1.4.4.2 事务特性:

原子性 :强调事务的不可分割.

一致性 :事务的执行的前后数据的完整性保持一致.

隔离性 :一个事务执行的过程中,不应该受到其他事务的干扰

持久性 :事务一旦结束,数据就持久到数据库

1.4.4.3 如果不考虑隔离性引发安全性问题:

脏读 :一个事务读到了另一个事务的未提交的数据

不可重复读 :一个事务读到了另一个事务已经提交的 update 的数据导致多次查询结果不一致.

虚幻读 :一个事务读到了另一个事务已经提交的 insert 的数据导致多次查询结果不一致.

1.4.4.4 解决读问题:设置事务隔离级别

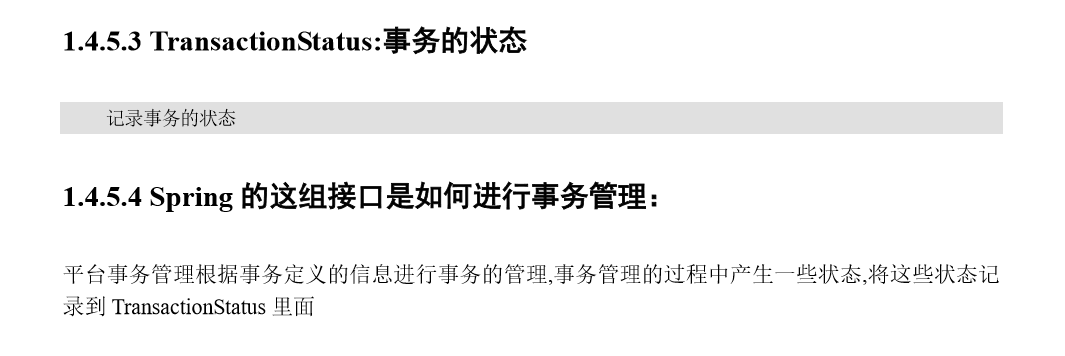
未提交读 :脏读，不可重复读，虚读都有可能发生

已提交读 :避免脏读。但是不可重复读和虚读有可能发生

可重复读 :避免脏读和不可重复读.但是虚读有可能发生.

串行化的 :避免以上所有读问题. 效率太低

Mysql 默认:可重复读 Oracle 默认:读已提交



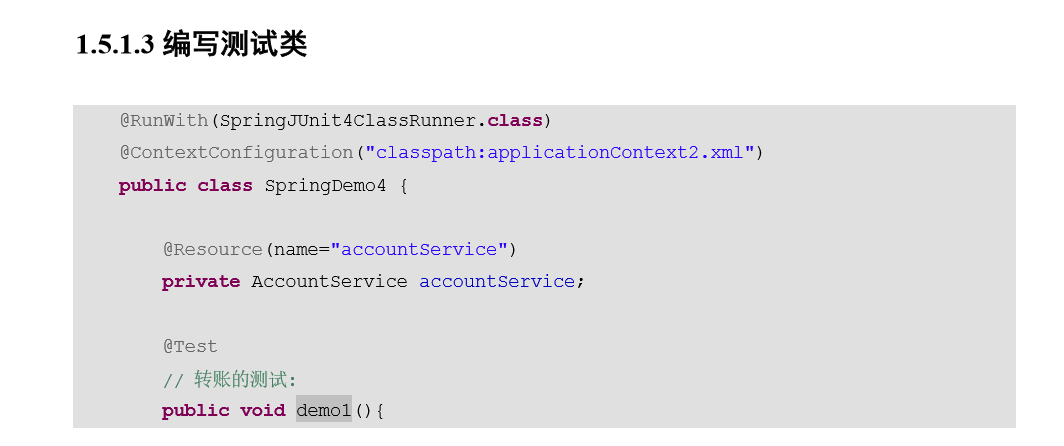


转钱：

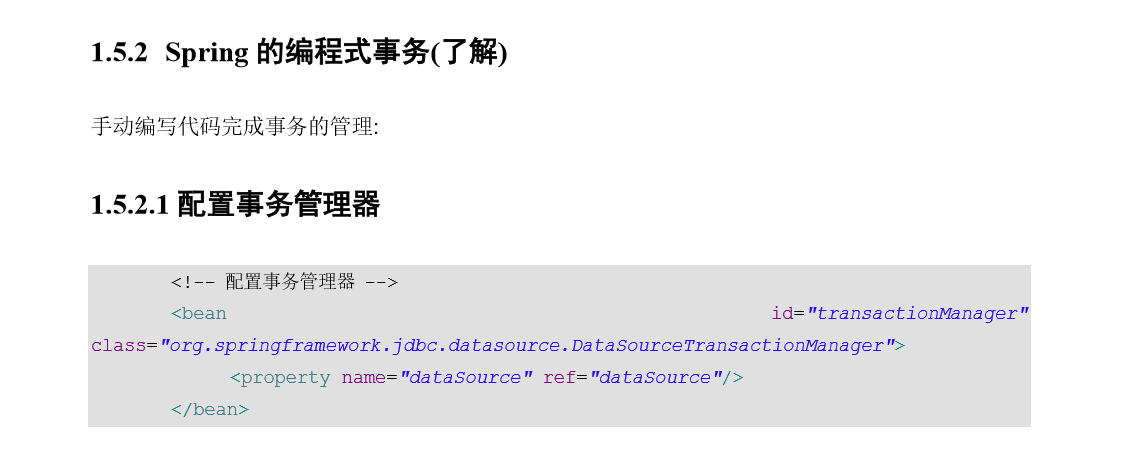




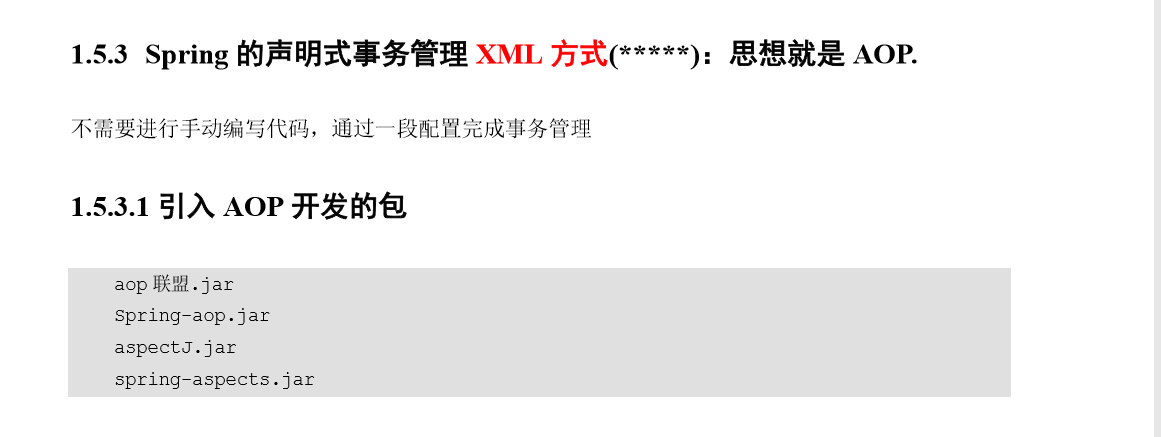








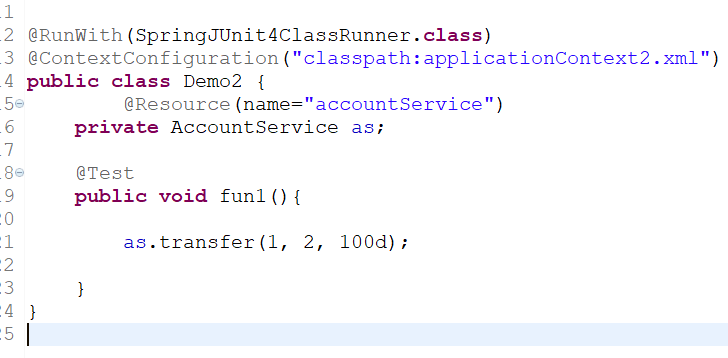






实例如下：

1. 测试类



2. 

其他不变

