Spring框架，可以解决对象创建以及对象之间依赖关系的一种框架。

且可以和其他框架一起使用；Spring与Struts, Spring与hibernate

(起到整合（粘合）作用的一个框架)

1） Spring Core spring的核心功能： IOC容器, 解决对象创建及依赖关系

2） Spring Web Spring对web模块的支持可以与struts整合,让struts的action创建交给spring,spring mvc模式

3） Spring DAO Spring 对jdbc操作的支持 【JdbcTemplate模板工具类】

4） Spring ORM spring对orm的支持, 既可以与hibernate整合，【session】,也可以使用spring的对hibernate操作的封装

5）Spring AOP 切面编程

6）SpringEE spring 对javaEE其他模块的支持

SpringIOC容器，是spring核心内容。

作用： 创建对象 & 处理对象的依赖关系

控制反转 将bean的对象交给spring容器进行管理

实现原理:反射---创建对象 解析xml,,demo4j

解析xml,获取bean calss地址,通过java反射,进行实例化对象,第三方返回对象

### 代理模式

:作用:对目标对象进行访问的方式(中介)

好处: 提高安全性

代理分为静态代理,动态代理:

静态代理: 产生代理类(中介)

动态代理: 不产生代理类,

分为:JDK动态代理 反射机制

Cglib动态代理 字节码

## 容器创建对象：

创建对象, 有几种方式：

1） 调用无参数构造器

2） 带参数构造器

3） 工厂创建对象

工厂类，静态方法创建对象

工厂类，非静态方法创建对象

AOP: Aspect Oriented Programming 面向切面编程。  
　　面向切面编程(也叫面向方面)：Aspect Oriented Programming(AOP),是目前软件开发中的一个热点。利用AOP可以对业务逻辑的各个部分进行隔离，从而使得业务逻辑各部分之间的耦合度降低，提高程序的可重用性，同时提高了开发的效率。  
　　AOP是OOP的延续，是（Aspect Oriented Programming）的缩写，意思是面向切面（方面）编程。  
　　主要的功能是：**日志记录，性能统计，安全控制，事务处理，异常处理**等等。  
　　主要的意图是：将日志记录，性能统计，安全控制，事务处理，异常处理等代码从业务逻辑代码中划分出来，通过对这些行为的分离，我们希望可以将它们独立到非指导业务逻辑的方法中，进而改  变这些行为的时候不影响业务逻辑的代码。

注意：AOP不是一种技术，实际上是编程思想。凡是符合AOP思想的技术，都可以看成是AOP的实现。

**Aop，** aspect object programming 面向切面编程

功能： 让关注点代码与业务代码分离！

**关注点,**

重复代码就叫做关注点；

**切面，**

关注点形成的类，就叫切面(类)！

面向切面编程，++]就是指 对很多功能都有的重复的代码抽取，再在运行的时候网业务方法上动态植入“切面类代码”。

**切入点，**

执行目标对象方法，动态植入切面代码。

可以通过切入点表达式，指定拦截哪些类的哪些方法； 给指定的类在运行的时候植入切面类代码。

# 一、事物的概述

## ⑴ 原子性（Atomicity）

　　原子性是指事务包含的所有操作要么全部成功，要么全部失败回滚，因此事务的操作如果成功就必须要完全应用到数据库，如果操作失败则不能对数据库有任何影响。

## ⑵ 一致性（Consistency）

　　一致性是指事务必须使数据库从一个一致性状态变换到另一个一致性状态，也就是说一个事务执行之前和执行之后都必须处于一致性状态。

　　拿转账来说，假设用户A和用户B两者的钱加起来一共是5000，那么不管A和B之间如何转账，转几次账，事务结束后两个用户的钱相加起来应该还得是5000，这就是事务的一致性。

## ⑶ 隔离性（Isolation）

　　隔离性是当多个用户并发访问数据库时，比如操作同一张表时，数据库为每一个用户开启的事务，不能被其他事务的操作所干扰，多个并发事务之间要相互隔离。

　　即要达到这么一种效果：对于任意两个并发的事务T1和T2，在事务T1看来，T2要么在T1开始之前就已经结束，要么在T1结束之后才开始，这样每个事务都感觉不到有其他事务在并发地执行。

　　关于事务的隔离性数据库提供了多种隔离级别，稍后会介绍到。

## ⑷ 持久性（Durability）

　　持久性是指一个事务一旦被提交了，那么对数据库中的数据的改变就是永久性的，即便是在数据库系统遇到故障的情况下也不会丢失提交事务的操作。

　　例如我们在使用JDBC操作数据库时，在提交事务方法后，提示用户事务操作完成，当我们程序执行完成直到看到提示后，就可以认定事务以及正确提交，即使这时候数据库出现了问题，也必须要将我们的事务完全执行完成，否则就会造成我们看到提示事务处理完毕，但是数据库因为故障而没有执行事务的重大错误。

# 四、传播七种行为

Spring中事务的定义：

Propagation（key属性确定代理应该给哪个方法增加事务行为。这样的属性最重要的部份是传播行为。）有以下选项可供使用：

* PROPAGATION\_required--支持当前事务，如果当前没有事务，就新建一个事务。这是最常见的选择。
* PROPAGATION\_SUPPORTS--支持当前事务，如果当前没有事务，就以非事务方式执行。
* PROPAGATION\_MANDATORY--支持当前事务，如果当前没有事务，就抛出异常。
* PROPAGATION\_REQUIRES 瑞快死\_NEW--新建事务，如果当前存在事务，把当前事务挂起。
* PROPAGATION\_NOT\_SUPPORTED--以非事务方式执行操作，如果当前存在事务，就把当前事务挂起。
* PROPAGATION\_NEVER 买佛--以非事务方式执行，如果当前存在事务，则抛出异常。