## Spring底层原理

### IOC的基本概念和原理：

基本概念：

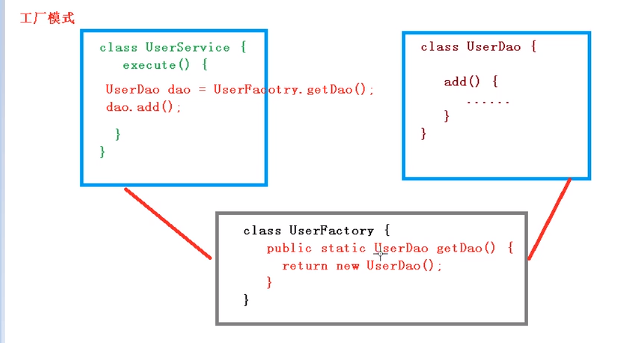
（1）控制反转，把对象的创建和对象之间的调用过程，都交给Spring管理；

（2）使用IOC的目的：为了降低耦合；

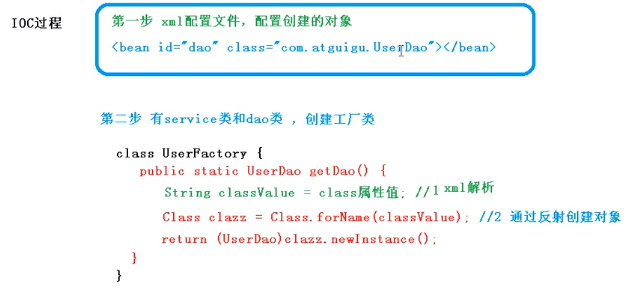
底层原理：主要用到了xml解析、工厂4模式、反射

工厂模式：假设UserService需要一个UserDao，就可以通过Factory来创建。假设不通过Factory，就需要在UserService中创建UserDao的对象。

目的是最大限度地降低耦合。



工厂模式和XML解析加上反射来得到对象：



进一步降低了耦合程度，此时，只需要修改xml文件就可以了。

IOC本质上就是一个容器，即对象工厂

IOC（接口）

1.IOC思想基于IOC容器完成，IOC容器底层就是对象工厂

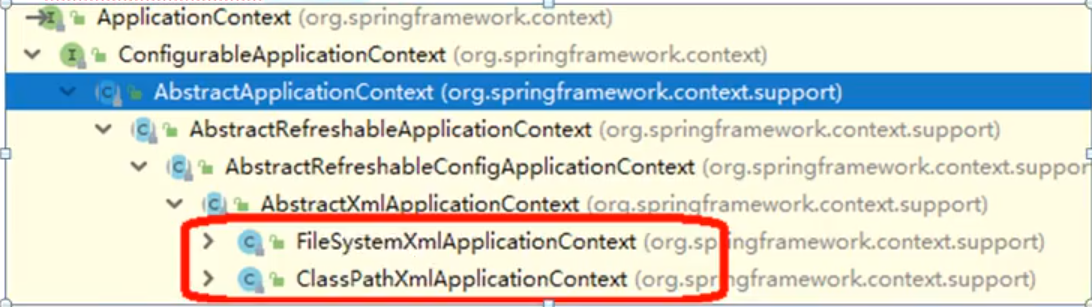
2.Spring提供IOC容器实现的两种方式：（两个接口）

（1）BeanFactory：IOC容器最基本的实现方式。是Spring内部使用的接口，一般不提供给开发人员使用；

（2）ApplicationContext：BeanFactory的子接口，提供的功能更加强大，一般由开发人员进行使用

（3）不同之处：BeanFactory 在加载配置文件的时候不会创建对象，在使用到对象的时候才会使用反射去创建；ApplicationContext在加载配置文件就会加载对象。

3.ApplicationContext的实现类：



上面的对应盘里面的路径，外面的对应src下的路径。

Bean管理操作的两种方式：

（1）基于xml配置文件方式

（2）基于注解的方式

注解：（1）注解是代码的特殊标记，格式@注解名称（属性名=属性值）；（2）注解可以在类上面，方法上面，属性上面；（3）使用注解的原因，简化xml配置，使用更简洁的方式配置xml。

Spring针对Bean管理创建对象提供的注解：

1）@Component

2）@Service

3）@Controller

4）@Repository

（各类注解一般用于各个层，但是只是建议，可以混用。功能都是一样的，都可以用来创建bean实例）

基于注解方式创建实例对象

操作步骤：

1.首先需要引入依赖：spring-aop依赖；

2.开启组件扫描（扫描那些类中的注解）；引入命名空间，在配置文件中引入component-scan；

3.在类中创建注解

基于注解方式实现属性注入：@Autowired（属性）；@Qualified（name）；@Resource（name/属性）；@Value

完全注解开发：

1.创建配置类，这个类需要在类的上面添加注解@Configuration，添加视图解析器；

2.在测试类中，再使用创建ApplicationContext 类，将这个类的class传进去。

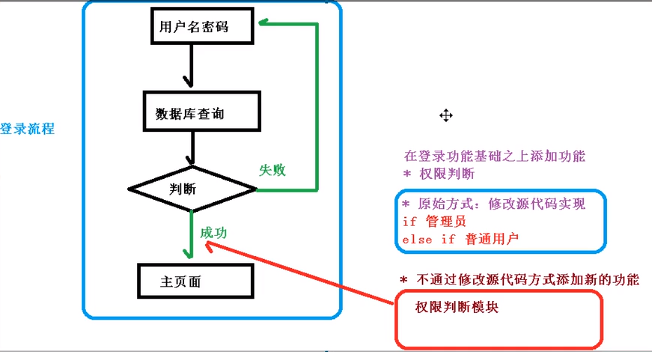
### AOP原理

概念：

1）AOP（面向切面编程）；是OOP的延续，可以使得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地耦合度降低，提高程序的可重用性，同时提高程序的开发效率；

2）通俗讲就是，不通过修改源代码的方式，在主干功能里添加新的功能；

3）以一个登陆的李子进行简要介绍。



底层原理：

1.AOP底层使用了动态代理方式来实现。

1）两种情况动态代理

有接口的情况，使用JDK动态代理：在接口的基础上，创建实现类，再创建接口实现类的代理对象，使用代理对象加强实现类；

无接口的情况，使用Cjlib实现。使用子类继承的方法来实现AOP：

动态代理的实现过程：

1）实用Proxy类中的一个newProxyInstance方法，其有三个参数：类加载器，增强方法所在类实现的接口数组，InvocationHandler接口，用于增强方法。

AOP术语：

1.连接点

类里面哪些方法可以被增强，这些方法叫做连接点；

2.切入点

实际被真正增强的方法，称为切入点。例如只增强某几个方法。

3.通知（增强）

（1）实际增强的逻辑部分称为通知（增强）

（2）通知的五种通知：前置（@Before），后置（@AfterReturning），环绕（@Around），异常（@AfterThrowing），最终（类似finally @After）；

4.切面

（1）是一个动作，把通知应用到切入点的过程。

基于AspectJ来实现AOP操作操作，

AspectJ不是Spring的组成部分，是独立的框架，一般是和Spring结合在一起，以进行AOP操作。

基于AsceptJ实现AOP操作：可以用xml配置，也可以使用注解实现。

切入点表达式：

（1）切入点表达式作用，指定对那个类的那个方法进行增强；

（2）语法结构：execution（[权限修饰符][返回类型][类全路径][方法名称]（[参数列表]））；

例子1：对com.njustxz.dao.BookDao中的add方法增强

execution(\* com.njustxz.dao.BookDao.add(..))

例子2：对com.njustxz.dao.BookDao中的所有方法增强

execution(\* com.njustxz.dao.BookDao.\*(..))

例子3：对com.njustxz.dao包中的所有类的所有方法增强

execution(\* com.njustxz.dao.\*\*(..))

具体操作：

1）引入依赖；

2）创建增强代理类；

3）Spring配置文件中开启注解组件扫描；

4）使用注解或者xml配置创建类与代理类对象；

5）在代理类上面使用@AsceptJ，表示这是个动态代理类；

6）在Spring配置文件中开启AsceptJ<aop:asceptl-autoproxy></aop:asceptl-autoproxy>

7）在增强方法上面使用注解（before）和切入点表达式，表示增强的类型和方法