Spring学习笔记

目录

[Spring学习笔记 1](#_Toc486511023)

[Spring\_day1 2](#_Toc486511024)

[1、Spring概念 2](#_Toc486511025)

[1.1、Spring核心两部分 2](#_Toc486511026)

[1.2、Spring一站式框架 2](#_Toc486511027)

[1.3、Spring版本 3](#_Toc486511028)

[2、Spring入门 3](#_Toc486511029)

[2.1、ioc底层原理 3](#_Toc486511030)

[2.2、ioc入门案例 5](#_Toc486511031)

[2.3、解决配置文件没有提示 6](#_Toc486511032)

[3、Spring的bean管理（xml配置文件） 7](#_Toc486511033)

[3.1、bean实例化三种方式 7](#_Toc486511034)

[3.2、bean标签常用属性 8](#_Toc486511035)

[3.3、属性注入方式介绍 9](#_Toc486511036)

[3.4、Spring注入属性（有参构造和set方法） 10](#_Toc486511037)

[3.5、spring注入对象类型属性（重点） 10](#_Toc486511038)

[3.6、P名称空间注入 11](#_Toc486511039)

[3.7、Spring注入复杂数据 11](#_Toc486511040)

[4、ioc和di区别 12](#_Toc486511041)

[5、spring整合web项目原理 13](#_Toc486511042)

[5.1、ServletContext对象 13](#_Toc486511043)

[5.2、监听器 13](#_Toc486511044)

[Spring\_day2 13](#_Toc486511045)

[1、 Spring的bean管理（注释） 13](#_Toc486511046)

[1.1、使用注解创建对象 15](#_Toc486511047)

[1.2、使用注解注入对象 15](#_Toc486511048)

[1.3、Xml和注解方式混合使用 15](#_Toc486511049)

[2、AOP 15](#_Toc486511050)

[2.1、Aop概述 15](#_Toc486511051)

[2.2、Aop底层原理 15](#_Toc486511052)

[2.3、Aop操作相关术语 15](#_Toc486511053)

[3、Spring的aop操作（基于aspectj的xml方式） 15](#_Toc486511054)

[4、Log4j 15](#_Toc486511055)

[5、Spring整合web项目演示 15](#_Toc486511056)

Spring课程安排

第一天Spring概念和ioc入门（ioc操作xml配置文件）

第二天spring的ioc操作（注解）和aop概念

第三天 jdbcTemplate 和Spring事物管理

第四天 SSH框架开发整合

# Spring\_day1

## 1、Spring概念

* Spring是一个轻量级开源框架，
* Spring是JAVAEE一个分层一站式框架

### 1.1、Spring核心两部分

* Aop：面向切面编程，扩展功能不是修改源代码来实现
* IOC控制反转、反转控制
  + 比如： 有一个类，类里面有方法（不是静态类），调用类里面的方法，创建类对象，使用对象调用方法，需要类对象过程，需要new出来对象

-把对象的创建不是通过new方式来实现，而是交给spring配置创建类对象

### 1.2、Spring一站式框架

* Spring在javaee三层结构中，每一层都提供不同的解决技术
  + Web层：springMVC
  + Service层：Spring的ioc
  + Dao层：spring的jdbcTemplate

### 1.3、Spring版本

* Hibernate 5.x
* Spring 4.x
  + 老项目都在用3
  + 新的都在用4

## 2、Spring入门

### 2.1、ioc底层原理

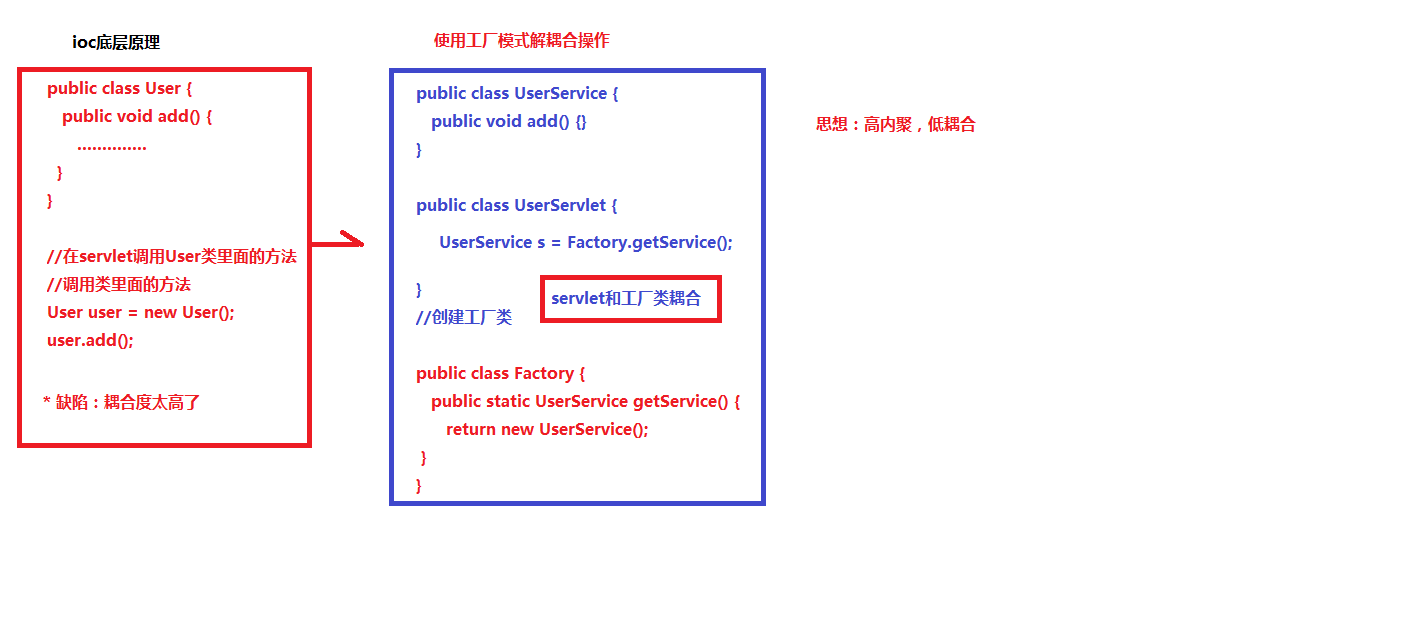
* IOC控制反转、反转控制
  + 比如： 有一个类，类里面有方法（不是静态类），调用类里面的方法，创建类对象，使用对象调用方法，需要类对象过程，需要new出来对象

-把对象的创建不是通过new方式来实现，而是交给spring配置创建类对象

‘’

* Spring的Ioc的操作
  + 把对象创建交给Spring进行管理
  + Ioc操作两部分
    - Ioc配置文件方式
    - Ioc的注解方式
* 底层使用的技术
  + Xml配置文件
  + Dom4j解决xml
  + 工程设计模式
  + 反射
* 画图分析ioc底层原理

原来的解决方案工程模式来实现创建类

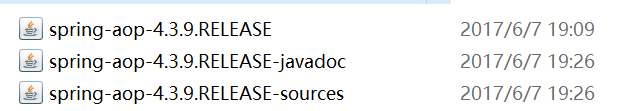


IOC模式来实现创建类

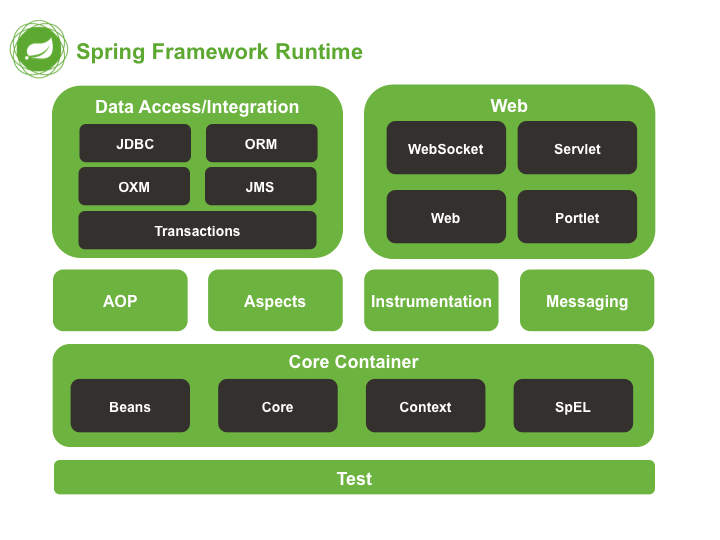
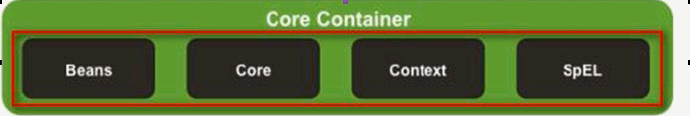
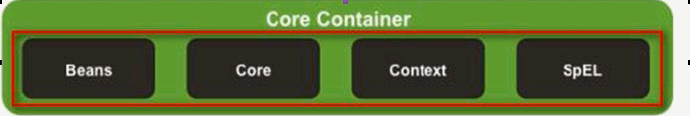


### 2.2、ioc入门案例

第一步：导入jar包



Lib中jar包中的结构



第二步：创建类，在类里面创建方法

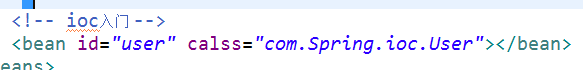
* Spring核心配置文件名称和位置不是固定的

-建议放到src下面，官方建议：applicationContext.xml

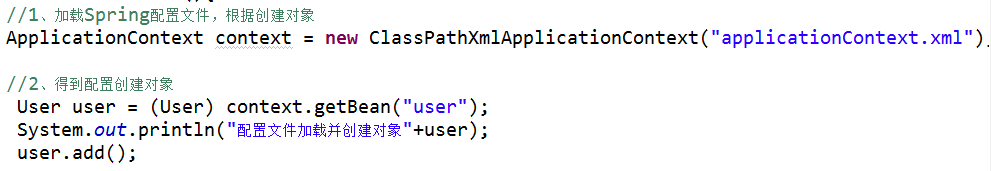
* 引入scheme约束

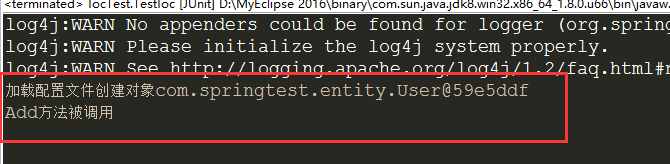
第三步：创建Spring配置文件，配置创建类

* 配置对象



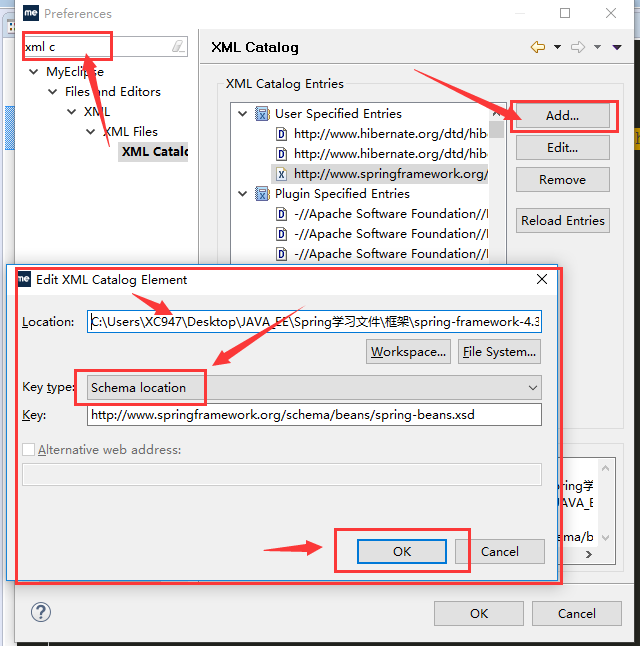
第四步：写代码测试Spring





### 2.3、解决配置文件没有提示

1. Spring引入scheme约束，把约束文件引入myeclipse



## 3、Spring的bean管理（xml配置文件）

### 3.1、bean实例化三种方式

1. 在Spring里面可以通过配置文件创建对象
2. bean实例化三种方式实现
   1. **使用类的无参数构造函数创建**

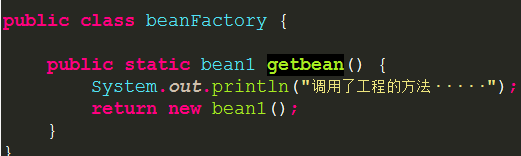
-如果类中不写则默认是无参数构造方法

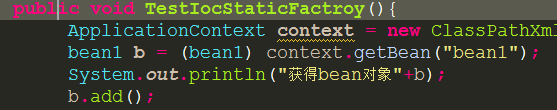
-如果没有无参数构造方法就会出现以下错误



* 1. **使用静态工厂创建**

-创建静态的方法，返回类对象





* 1. **使用实例工厂创即**

-创建不是静态方法，返回类对象（利用类的对象创建）

### 3.2、bean标签常用属性

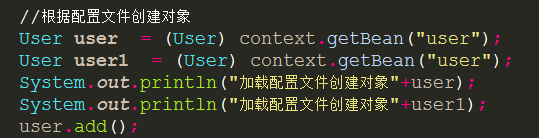
1. Id属性：起名称，id属性值名称任意命名

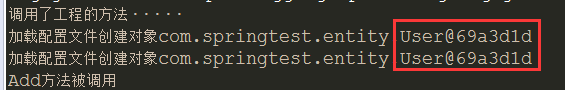
-id属性值，不能包含特殊符号”\_”也不例外

-根据id值得到配置对象

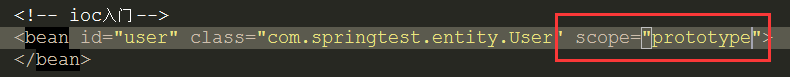
1. Class属性：创建对象所在的类的全路径
2. Name属性：name的功能和id属性一样，区别在于name可以包括特殊符号
3. Scop属性：

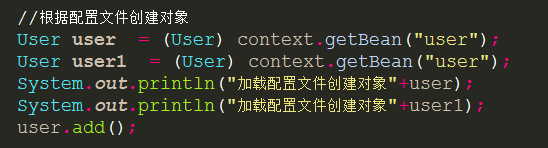
**-singleton：单例对象 （默认是单例对象）**

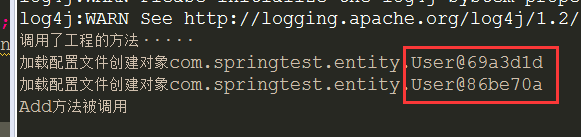




-**prototype ：多例 ：类似action，单例的话就会共享某些隐私数据**







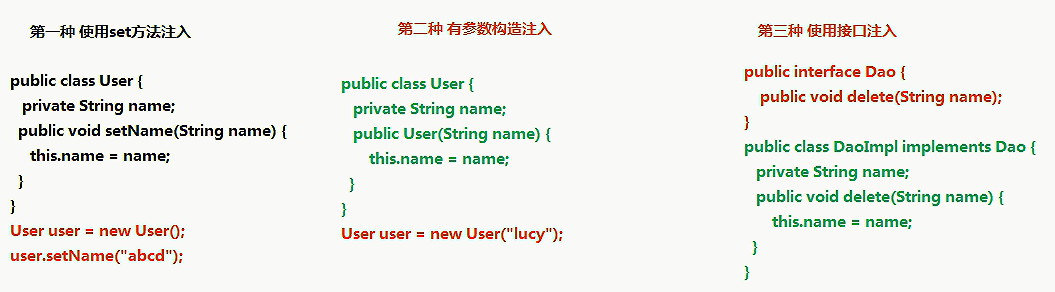
**-requset ：创建对象把对象放到request域里面**

**-session：创建对象把对象放到session域里面**

**-GlobalSession：创建对象把对象放到globalSession域里面**

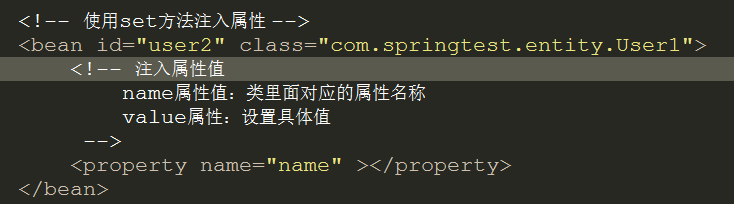
### 3.3、属性注入方式介绍

1. 创建对象时，向对象类的里面属性注入值
2. 属性注入的三种方式介绍
   1. 使用set方法设置值
   2. 使用有参数构造函数注入
   3. 使用接口注入

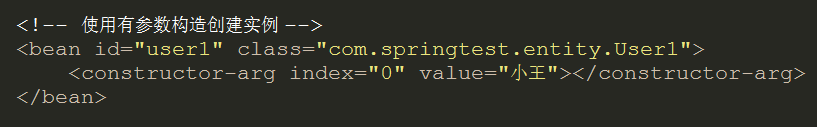


### 3.4、Spring注入属性（有参构造和set方法）

1. 在Spring框架里面只支持前面两种方法
   1. Set方法（重点）

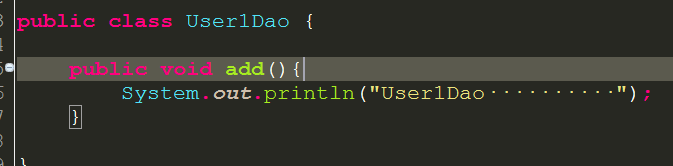


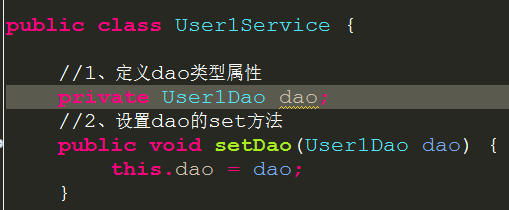
* 1. 有参数构造注入



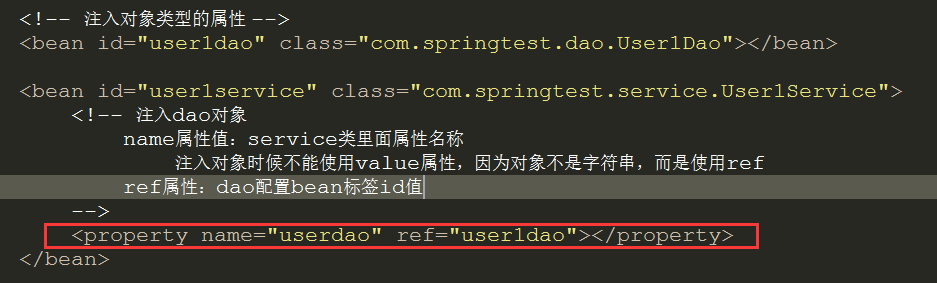
### 3.5、spring注入对象类型属性（重点）

1. 创建service类和dao类
   1. 在service得到dao对象
2. 具体实现过程
   1. 在service里面把dao作为类型属性
   2. 生成dao类型属性的set方法



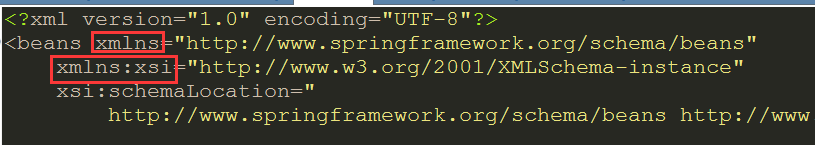


* 1. 写配置文件

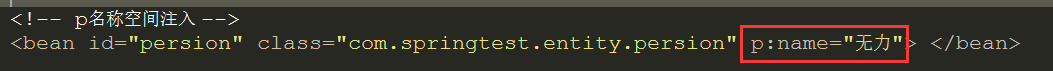


### 3.6、P名称空间注入

在一个标签中不能写同一个属性 ，使用p名称空间可以写同属性名的，但是其原理是在原属性上加“：别名”



Set方法设置值



### 3.7、Spring注入复杂数据

1. 数组
2. List集合
3. Map集合
4. Properties类型



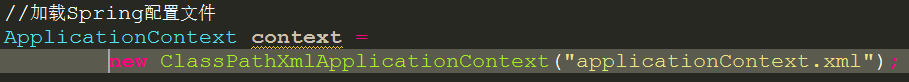


## 4、ioc和di区别

1. IOC：控制反转，把对象创建交给Spring进行配置
2. DI ：依赖注入，向类里面的属性中设置值
3. 关系：依赖注入不能单独存在的，需要在IOC基础之上完成操作

## 5、spring整合web项目原理

1. 加载Spring核心配置文件



New对象，功能可以实现，效率低

1. 实现思想：把加载配置文件和创建对象过程，在服务器启动的时候完成
2. 实现原理：
   1. ServletContext对象
   2. 监听器
   3. 具体原理如下：

·--在服务器启动的时候，为每个对象创建ServletContext对象

·--在ServletContext对象创建时候，使用监听器可以具体到ServletContext对象在什么时候创建

·--使用监听器监听到ServletContext对象创建时候

--加载spring配置文件，把配置文件配置对象创建

·--把创建出来的对象放到ServletContext域对象里面（setAttribute方法）

-·-获取对象时候，到ServletContext域得到（getAttribute方法）

### 5.1、ServletContext对象

### 5.2、监听器

# Spring\_day2

主要内容：

Spring的bean管理

AOP

Spring的aop操作（基于aspectj的xml方式）

Log4j

Spring整合web项目演示

## Spring的bean管理（注释）

1. 代码里面特殊标记，使用注解可以完成用功能
2. 注解写法：“ @ + 注解名称（属性名 = 属性值）”
3. 注解使用范围： ① 类上边、② 方法上边、③ 属性上边
4. Spring注解开发准备工作

--导入Spring基本jar包

--导入aop的jar包

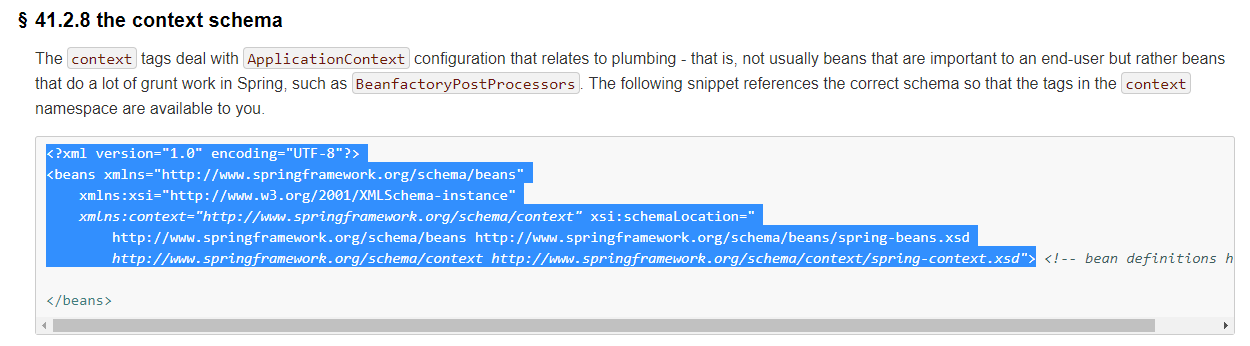


--创建对象、实体

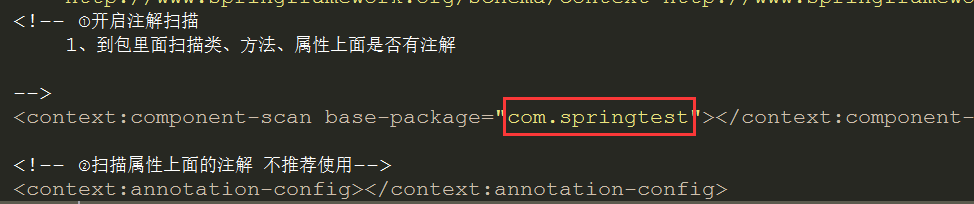
--创建xml配置文件

引入约束beans

引入新的约束

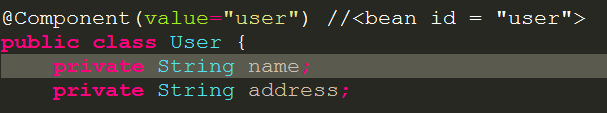


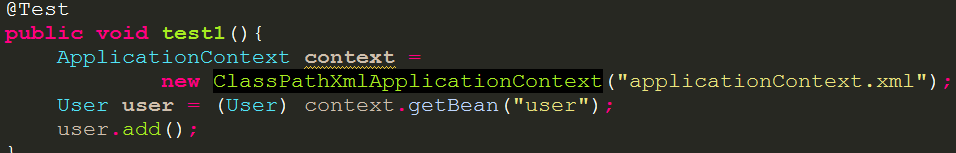
--配置xml文件，引入扫描注解标签



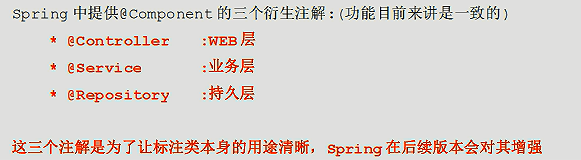
### 使用注解创建对象

1. 在创建对象的类上边使用注解



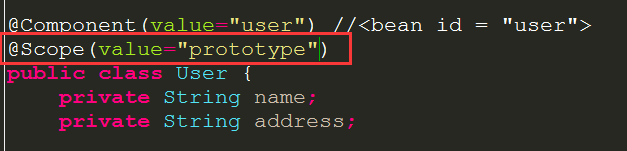


1. 创建对象有四种注解方式



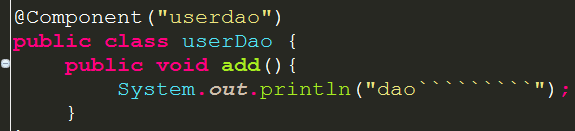
四种注解的功能都是一样的，都创建对象

1. 创建多实例对象



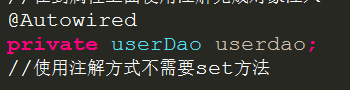
### 使用注解注入对象

1. 创建service类，创建dao类，在service得到dao对象
   1. 创建dao和service对象

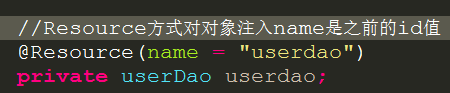


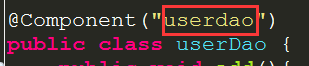


* 1. 在service类里面定义dao类型属性



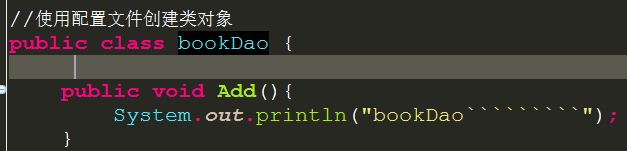
@autowored自动注入：根据类名找类对应的对象

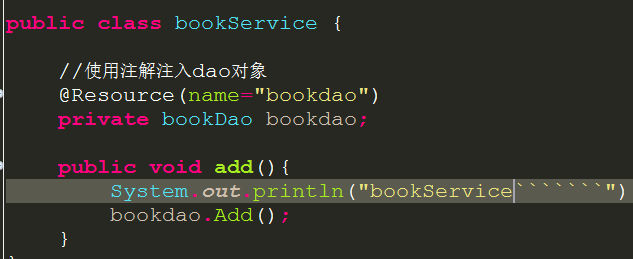


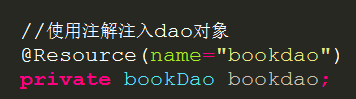


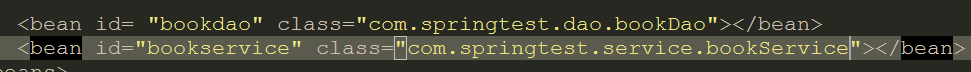
@Resource注入：根据设置好的值，查找对应的对象

### Xml和注解方式混合使用









总结：

1. 创建对象操作使用配置文件方式实现
2. 注入属性的操作使用注解方式实现

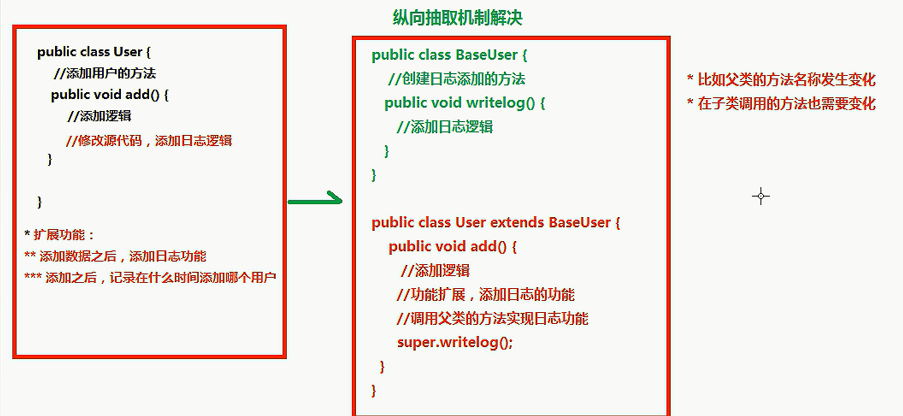
## 2、AOP

### 2.1、Aop概述

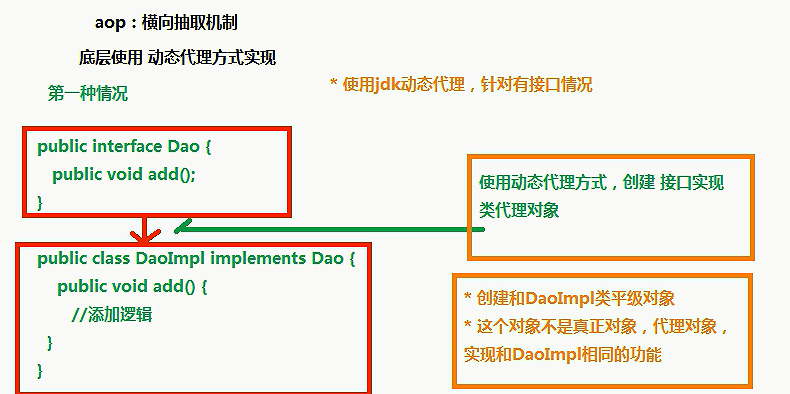
1. AOP：面向切面编程、或者面向方面编程
2. AOP采用横向抽出机制，取代了传统纵向继承体系重复代码制代码

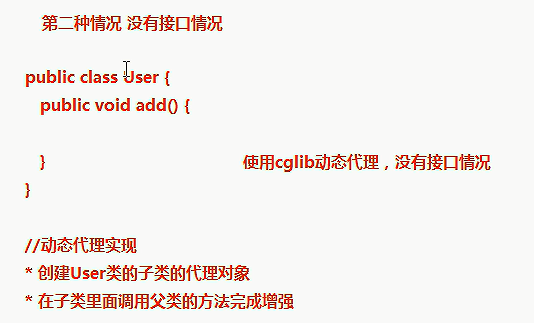
### 2.2、Aop底层原理

* + 纵向机制就是继承的关系：User1 extends User



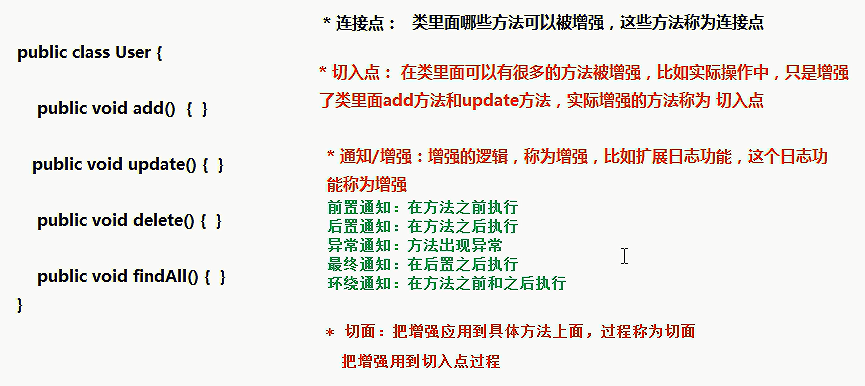
* + 采用横向机制





### 2.3、Aop操作相关术语

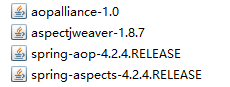
Joinpoint(连接点):所谓连接点是指那些被拦截到的点。 在 spring 中,这些点指的是方法,因为 spring 只  
持方法类型的连接点.  
Pointcut(切入点):所谓切入点是指我们要对哪些 Joinpoint 进行拦截的定义.  
Advice(通知/增强):所谓通知是指拦截到 Joinpoint 之后所要做的事情就是通知.通知分为前置通知,后置  
知,异常通知,最终通知,环绕通知(切面要完成的功能)  
Introduction(引介):引介是一种特殊的通知在不修改类代码的前提下, Introduction 可以在运行期为类  
态地添加一些方法或 Field.  
Target(目标对象):代理的目标对象  
Weaving(织入):是指把增强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在期织入  
Proxy（代理） :一个类被 AOP 织入增强后， 就产生一个结果代理类  
Aspect(切面): 是切入点和通知（引介） 的结合



## Spring的aop操作

AspectJ是一个面向切面编程的框架，它是java的扩展

1. 再Spring里面进行aop操作，使用aspertj实现
   1. AspectJ不是Spring一部分，和Spring一起使用进行aop操作
   2. Spring2.0以后新增了两对AspectJ支持
2. 使用aspectj实现aop两种方式
   1. 基于aspectJ的配置文件方式
   2. 基于aspectJ的注解方式
3. 准备工作;
   1. 导入相关的jar包



* 1. 创建一个Spring核心配置文件，导入aop约束

1. 使用表达式配置切入点
   1. 切入点：时间增强的方法
   2. 常用表达式
      * Execution（ <访问修饰符> ？ <返回类型> <方法名>（<参数>）<异常> ）

Eg：1、execution（\* com.xzq.action.Book.add(…) ）对action包中的Book类add的方法增强

2、execution（\* com.xzq.action.Book\*）对action包中的Book类所有方法增强

3、execution（\* \*.\*(..)）所有包，所有类的方法增强

Xml操作如下：



## 4、Log4j

1. 通过log4j可以看到程序运行过程中更详细的信息
   1. 经常使用log4j查看日志
2. Log4j怎么去使用呢？
   1. 导入log4j的jar包
   2. 复制log4j的配置文件，复制到src目录下面



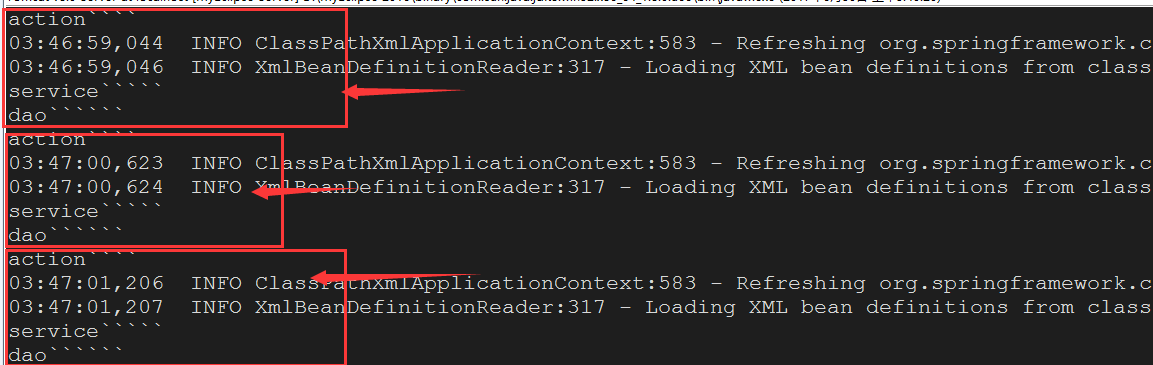
1. 设置日志级别



* 1. Info：看到基本信息
  2. Debug：看到详细信息

## 5、Spring整合web项目演示

1. 演示问题：
   1. Action调用service，service调用dao

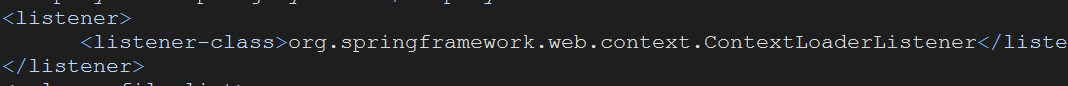


每次访问action时候，都会加载Spring配置文件

1. 解决方案：
   1. 在服务器启动时候，创建对象加载配置文件
   2. 底层使用监听，ServletContext 对象
2. 在Spring里面不需要我们自己写代码，已经帮封装好了
   1. 封装了监听器，我们只需要配置监听器皆可以了
   2. 配置监听器前要做的事情：导入整合jar包



* 1. 配置web.xml文件



* 1. 指定Spring配置文件的位置（不指定会报错）

