# Spring

Spring概念：

spring是开源轻量级框架

spring的核心：aop ioc

aop：面向切面编程 扩展功能不是通过修改代码实现  
 ioc：控制反转 创建对象 不是使用new一个对象 而是使用spring的配置创建一个对象

spring是一站式框架

spring在javaee三层框架中 每一层都提供了不同的解决技术：

web层：springMvc

service层：spring – ioc

dao层：通常都与mybatis整合

spring的ioc操作：

ioc的两部分操作：

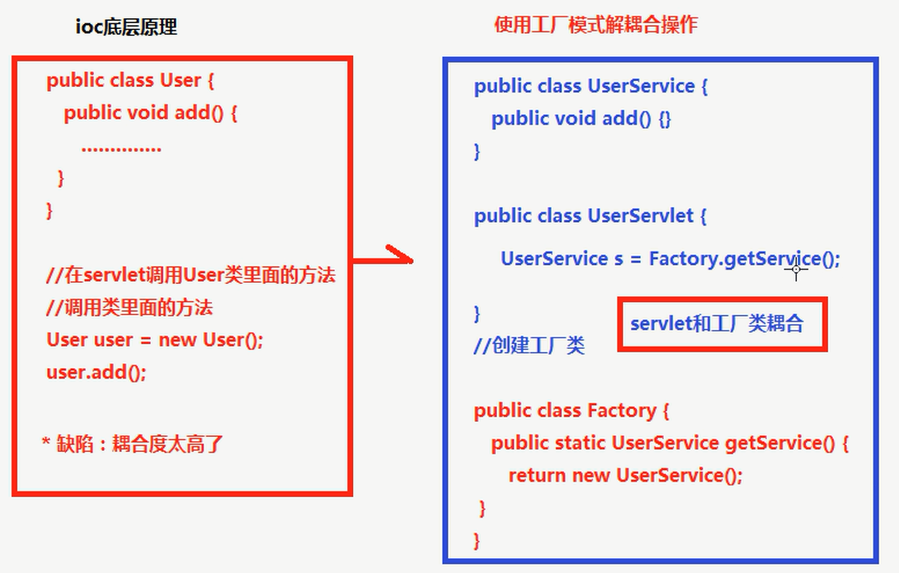
ioc注解方式

ioc配置方式

ioc的底层原理：

xml配置文件 dom4j解决xml 工厂设计模式 反射

ioc的实现原理：（使用ioc的原因之一是保持低耦合）



之前的设计方式耦合度过高

应该采用 高内聚低耦合的设计方式

配置spring：

导入jar之后

使用applicationContent.xml 作为全局配置文件

写入约束文件 schema

spring的bean管理：

bean实例化的方式：

在spring通过配置文件创建bean实例化

使用静态工厂创建（重点）

如果class里面 没有无参构造器 那么会报错！！

bean标签的常用属性：

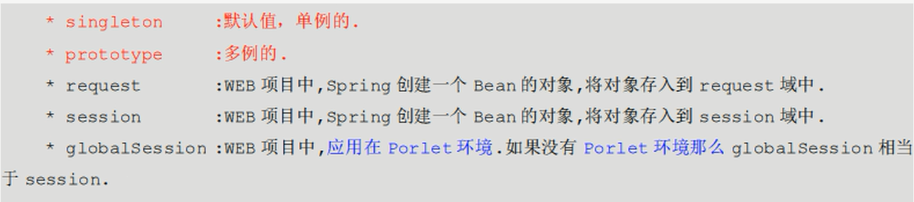
id： 相当于起了一个名字 可以任意命名

class： 创建对象所在的全路径

name： 跟id相似 但是id的属性值里面不能包含特殊符号（没人用 …）

scope： Bean的作用范围

Scope属性:



前面两个重要

singleton：默认值 单例

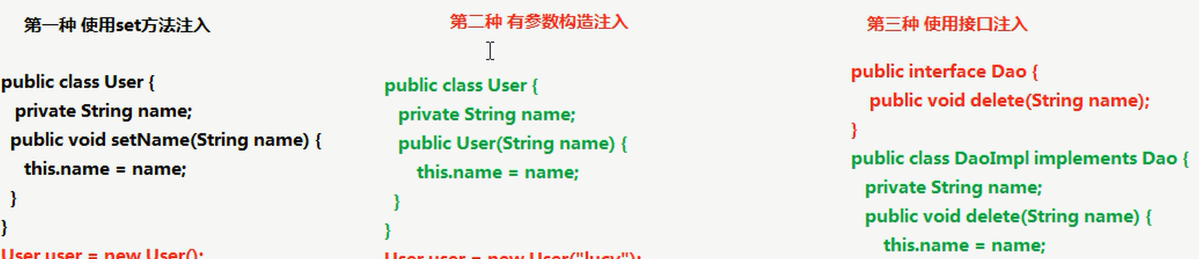
创建多个相同的bean对象 都是指向同一个对象

prototype： 多实例

创建多个相同的bean对象 指向不同内存地址

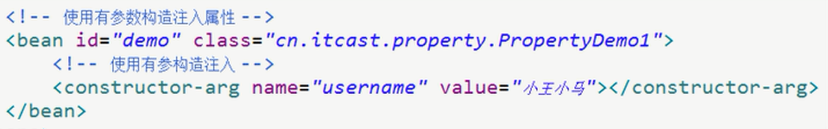
属性的注入：

创建一个新对象时 向类里面的属性设置值

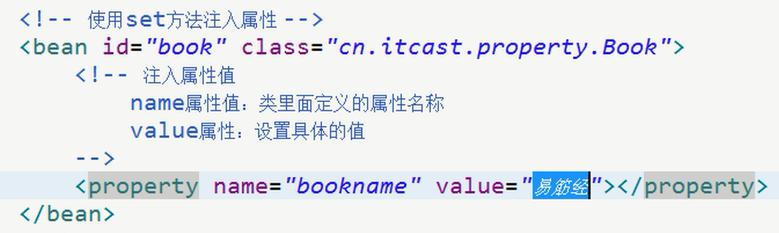


在spring只支持 set方法注入

有参构造器注入



使用set方法添加：



只要class中有set方法 就可以新加property标签

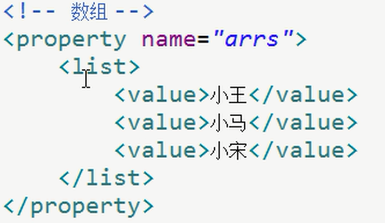
property标签中 name属性是class里的参数名 value是要传入的值

注入对象类型属性：

向class中的类属性 注入对应的class实例



复杂加载：

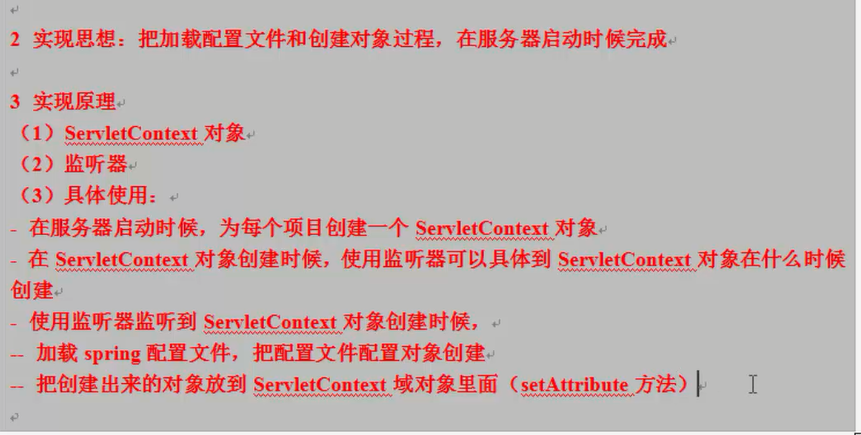








对于初始化配置文件：



ioc注解方式 ：

注解：代码里的特殊标识

注解可以使用在 方法 类 属性 上

spring注解的开发：

导入jar包

创建一个类 创建方法

创建spring配置文件 引入约束

在spring配置中 开启注解扫描

<context:component-scan base-package=”要扫描的全路径”> 开启注解扫描 配置比较方便

注解：

Component

Controller web 层

Service 业务层

Repository 持久层

这四个注解的功能都是用来操作创建对象

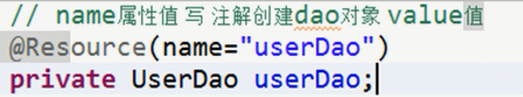


注解输入属性（两种方法）：



@Autowired 表示自动装配属性值

通过 属性 的class类型去找到对应的对象



@Resource（name=”要指定对象的value（使用注解方式创建的value值）”）

需要指定对象

配置文件 注解混合使用：



AOP： 面向切面编程

AOP原理：横向处理机制

连接点： 类里面那些方法可以被增强 叫做连接点

切入点： 实际增强的方法叫做切入点

通知/增强： 实际增强的逻辑 实际扩展的功能

例如扩展了日志功能 日志就叫增强

前置通知：在方法之前执行

后置通知 异常通知 最终通知

环绕通知：方法之前 之后都执行

切面：（重要）

把增强应用到具体的方法上

Spring 里的 AOP 操作：

在Spring进行aop操作

AspectJ 不是 spring的一部分

使用aspectj进行aop操作

使用aspectj实现aop的方式：（2种）

AOP操作：

导入jar包

使用表达式 配置切入点

execution(\* cn.lxa.book.add(..))方法的全路径

execution(\* cn.lxa.book.\*(..)) bookclass中的所有方法

execution(\* save\*(..))表示对所有有save的方法增强aspectj的aop操作：



log4j：



aspect注解aop操作：

jdbcTemplate

## RequestMapping简介：

@RequestMapping

RequestMapping是一个用来处理请求地址映射的注解，可用于类或方法上。用于类上，表示类中的所有响应请求的方法都是以该地址作为父路径。

RequestMapping注解有六个属性，下面我们把她分成三类进行说明。

### 1、 value， method；

value：     指定请求的实际地址，指定的地址可以是URI Template 模式（后面将会说明）；

method：  指定请求的method类型， GET、POST、PUT、DELETE等；

**还有一个注意的**，@RequestMapping的默认属性为value，所以@RequestMapping(value="/example")和@RequestMapping("/example")是等价的。

### 2、 consumes，produces；

consumes： 指定处理请求的提交内容类型（Content-Type），例如application/json, text/html;

produces:    指定返回的内容类型，仅当request请求头中的(Accept)类型中包含该指定类型才返回；

### 3、 params，headers；

params： 指定request中必须包含某些参数值是，才让该方法处理。

headers： 指定request中必须包含某些指定的header值，才能让该方法处理请求。