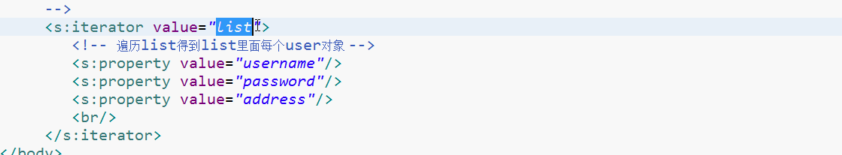
向值栈中放list集合



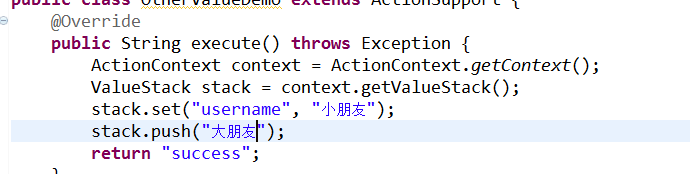


从值栈中获取数据

1. 获取字符串
2. 获取对象
3. 获取list集合

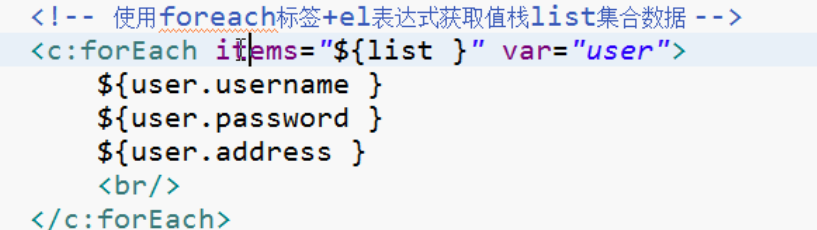


其他操作



1. 使用set方法向值栈放数据，获取
2. Push方式向值栈放数据，获取





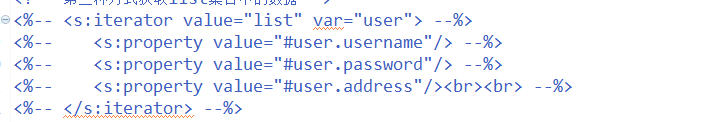
# EL表达式获取值栈数据（为什么）

1. EL表达式获取域对象值
2. 向域对象里面放值使用setAttribute方法，取值用getAttribute方法
3. 底层增强request对象里面的getAttribute方法
   1. 首先从request域对象取值，如果取到，直接返回
   2. 如果从域对象里面取不到值，到值栈中把值取出来，把值放到域对象里面

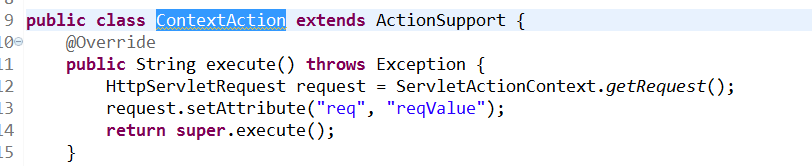
# OGNL的#、%使用

#使用

1. 使用#获取context里面的数据



1. 演示#操作
2. 向request域放值



1. 
2. 在页面中使用ogal获取

3、%使用



4、基本案例的实现



# Struts2拦截器概述

1. struts2是框架，封装了很多功能，struts2里面封装的功能都是在拦截器里面
2. struts里面封装了很多功能，有很多拦截器，不是每次这些拦截器都执行，每次执行默认的拦截器
3. struts里面默认拦截器位置



1. 拦截器在什么时候执行？
2. 在action对象创建之后，方法执行之前

# 拦截器底层原理

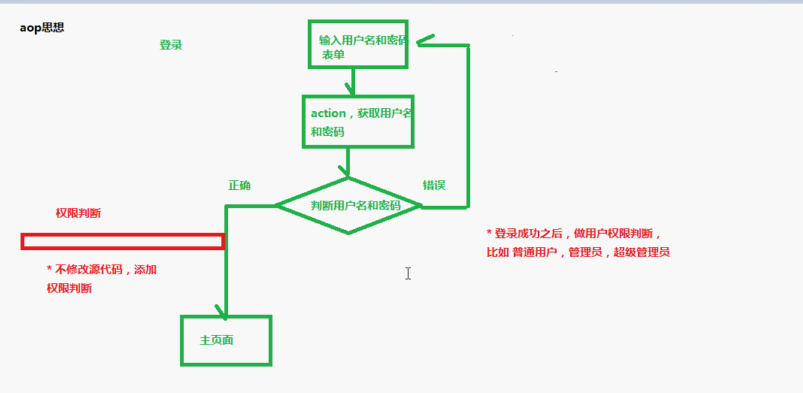
1. 拦截器底层使用的两个原理

第一个aop思想

（0）在spring里面把aop做更深层分析

1. 文字描述

Aop是面向切面（方面）编程，有基本功能，扩展功能，不通过修改源代码方式扩展功能



第二个思想：责任链思想

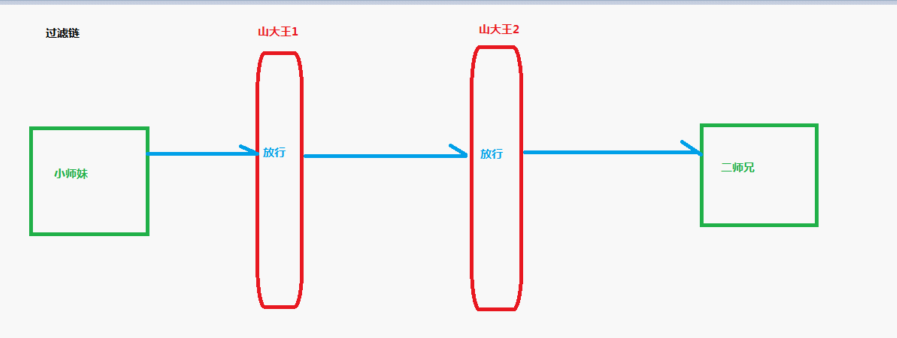
1. 在java中有很多设计模式，责任链模式是其中一种
2. 责任链和过滤链很相似

责任链模式：

要执行多个操作，有添加、修改、删除三个操作

首先执行添加操作，添加操作执行之后做类似于放行操作，执行修改操作，修改操作执行之后做类似于放行操作，执行删除操作

过滤器：一个请求可以有多个过滤器进行过滤，每个过滤链只有做放行才能到下一个过滤器



1. aop思想个责任链模式如何应用到拦截器里面
2. 文字描述

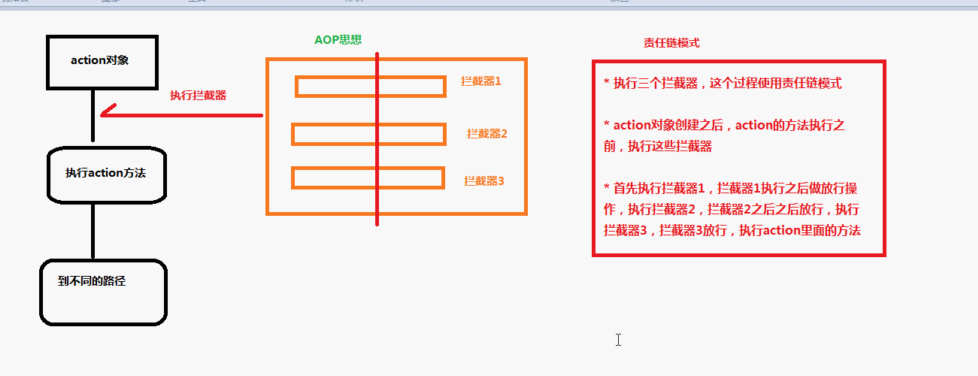
拦截器在action对象创建之后，action方法执行之前执行

在action方法执行之前执行默认拦截器，执行思想aop思想，在action里面并没有直接调用拦截器的方法，使用配置文件方式进行操作

在执行拦截器十九，执行很多拦截器，这个过程使用责任链模式

假如执行三个拦截器，执行拦截器1，执行拦截器1之后做放行操作，执行拦截器2，执行拦截器2之后放行，执行拦截器3，执行拦截器3之后放行，执行action里面方法

1. 画图分析



1. 查看源代码、



# 重要概念

1. 过滤器和拦截器区别
2. 过滤器：过滤器理论上可以过滤任何内容，比如html、jsp、servlet、图片路径
3. 拦截器：拦截器可以拦截action
4. Servlet和action区别
5. Servlet:默认是第一次访问时候创建，创建一次，单实例对象
6. Action每次访问的时候创建，创建多次，多实例对象

# 自定义拦截器

1. 在struts2里面有很多拦截器，这些拦截器是struts封装的功能，但是在实际的开发中，struts里面的拦截器可能没有我们想要的功能，这个时候需要自己写拦截器实现功能
2. 拦截器结构

（1）查看源代码看它的结构



（2）开发中，建议使用另外一种方式

写类，继承MethodFilterInterceptor类实现

让action里面某个方法不进行拦截

1. 让拦截器和action有关系

不是在action调用拦截器的方法，而是通过配置文件方式让建立关系

# 案例：自定义登录拦截器

1. 需求：在项目中，有很多action的超链接，实现只有登录状态，才能点击action的超链接实现功能，如果不是登录状态，点击action超链接返回登录界面
2. 登录状态，使用session域对象实现
3. 登录成功之后，把数据放到session里面
4. 判断session是否有值，可以知道是否是登录状态
5. 实现登录的基本功能



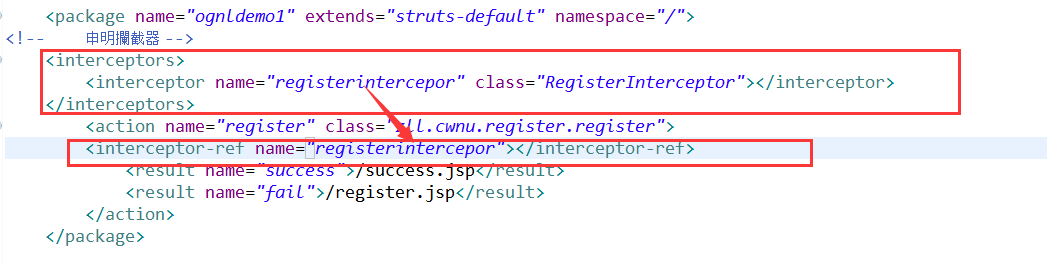
1. 查询数据库判断用户名和密码
2. 添加登录拦截器
3. 判断是否登录，判断session里面是否有名称是username的值
4. 拦截器实现过程

第一步：创建一个类，继承MethodFilterInterceptor

第二步：重写MethodFilterInterceptor类里面的方法写拦截器逻辑

第三步：配置action和拦截器的关系（注册拦截器）

1. 在要拦截的action标签所在的package标签里面申明拦截器
2. 在具体的action标签里面使用申明的拦截器



1. struts里面执行很多默认的拦截器，但是如果在action里面配置了自定义拦截器，默认的拦截器就不会执行了

解决：把默认拦截器手动的使用一次



1. 配置的拦截器会对action里面所有的方法都进行拦截
2. 在action里面有login方法，这个方法不需要拦截，如果这个方法都拦截；问题是：永远登录不进去了
3. 解决：让login方法不进行拦截

直接通过配置的方式，让action中的某些方法不进行拦截





# Struts2的常用的标签库

1. struts2标签使用isp页面中
2. s:property:和ognl表达式在jsp页面中获取值栈数据
3. s:iterator:获取值栈list集合数据，表示list集合
4. s:debug：查看值栈结构和数据

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

# Spring概念

1. Spring是轻量级框架
2. Spring核心主要两个部分
3. aop:面向切面编程，扩展功能不是修改源代码实现
4. ioc：控制反转

比如有一个类，在类里面有方法（不是静态方法），调用类里面的方法，创建类的对象，使用对象调用方法，，创建类对象的过程，需要new出来对象

把对象的创建不是通过new方式实现，而是交给spring配置创建类对象

1. spring是一站式框架
2. 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. spring版本
   1. hibernate5.x
   2. spring4.x

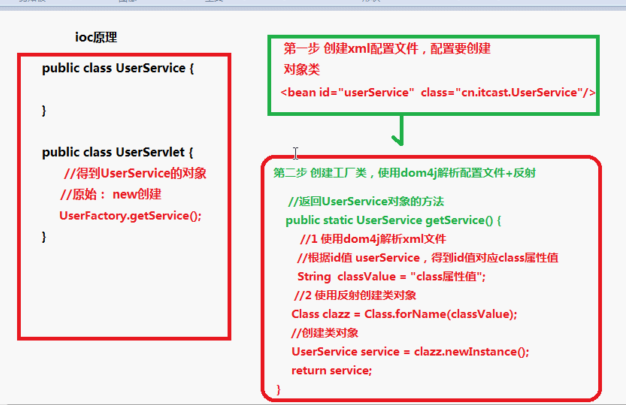
## spring的ioc

1. 把对象的创建交给spring进行管理
2. Ioc操作两部分
   1. Ioc的配置文件方式
   2. Ioc的注解方式

## Ioc底层原理



1. xml配置文件
2. dom4j解析xml
3. 工厂设计模式
4. 反射

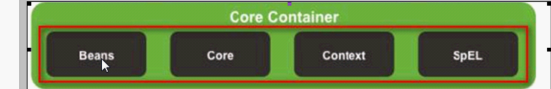


# Ioc入门案例

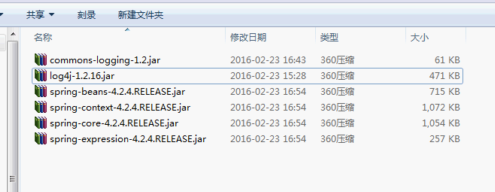
第一步：导入jar包

* + 1. 解压jar包的特点，都有三个jar包





* + 1. 做spring最基本功能时候，导入四核核心的jar包就可以了
    2. 导入支持日志输出的jar包



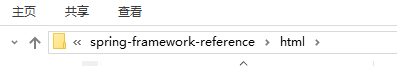
第二步：创建一个类，在类里面创建方法

第三步：创建spring配置文件，配置创建类

1. 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

1. 引入schema约束



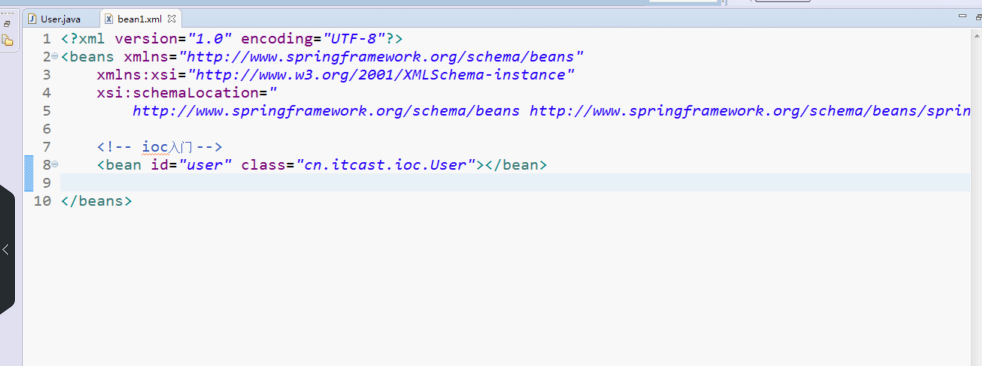
<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="

http://www.springframework.org/schema/beans <http://www.springframework.org/schema/beans/spring-beans.xsd>">

1. 配置操作



第四步：写代码测试对象创建

1. 这段代码在测试中使用



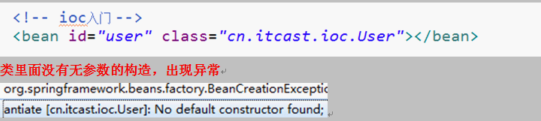
## 解决配置文件不能提示的问题

1. spring引入schema约束。把约束文件引入到eclipse中

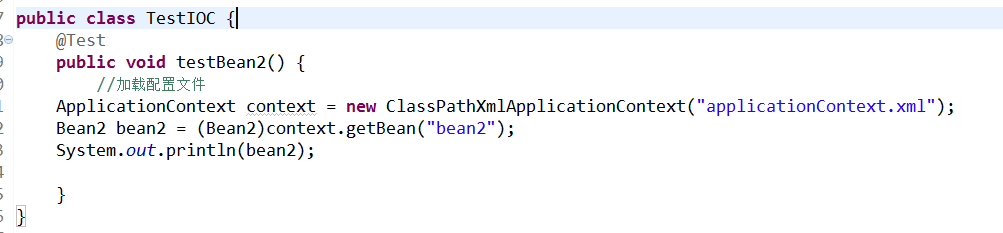
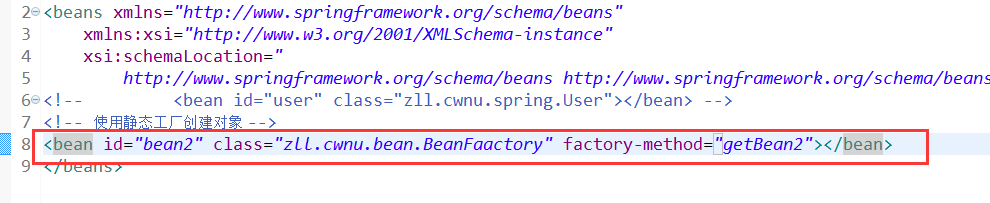
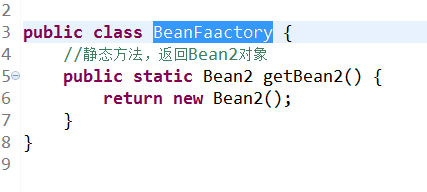
Spring的bean管理（xml方式）

1. bean实例化的方式
   1. 在spring里面通过配置文件创建对象
   2. Bean实例化三种方式实现

第一种：使用类的无参数构造创建（重点）

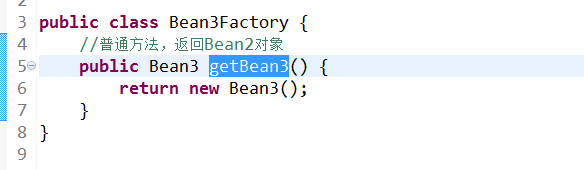


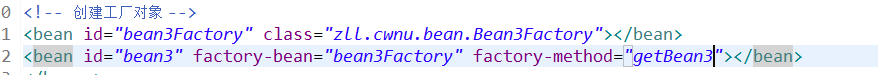
第二种：使用静态工厂创建

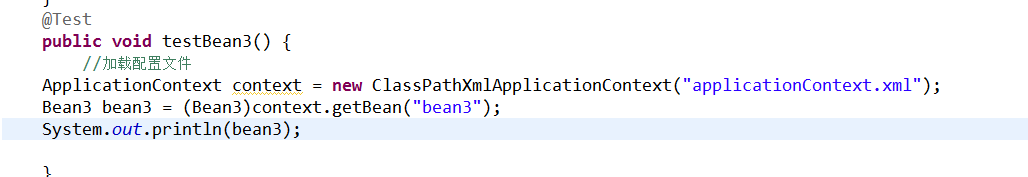
1. 创建静态方法，返回类实例
2. 
3. 

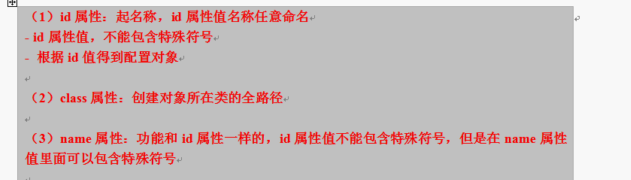
第三种：使用实例工厂创建

1. 创建不是静态方法，返回类的对象









1. scope属性

singleton:默认值，单例

prototype:多例

request:创建对象把对象放到request域中

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里面

# 属性的注入

1. 在创建对象的时候，向类里面的属性设置值
2. 属性注入方式（三种）
3. 使用set方法注入
4. 使用有参数构造
5. 使用接口注入

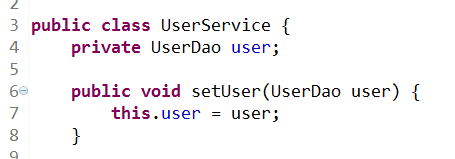


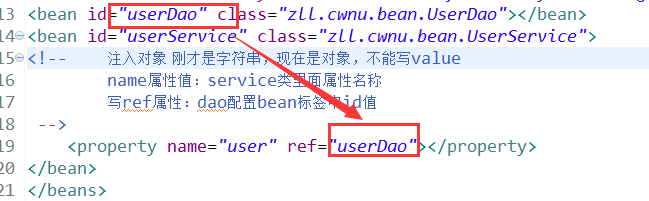
1. 在spring框架中，支持前两种方式
2. set方式注入
3. 有参数构造注入



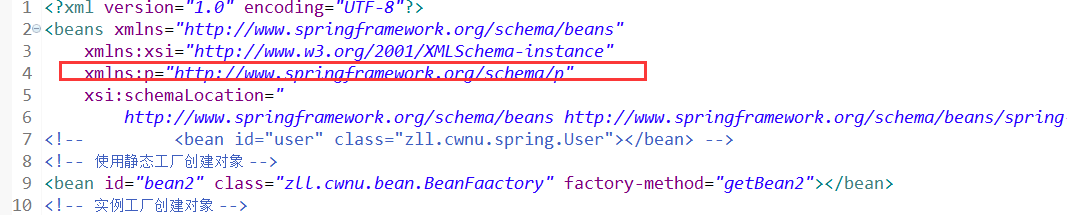
# 注入对象类型属性（重点）

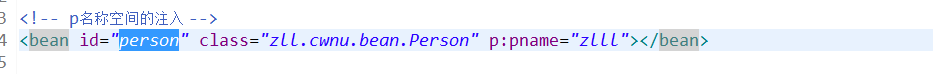
1. 创建service类和dao类
2. 具体实现过程
3. 在service里面把dao作为类型属性
4. 生成dao类型属性的set方法





# P名称空间注入





# 注入复杂类型属性

1 数组

2 list集合

3 map集合

4 properties类型





# 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，需要在IOC基础之上完成操作

# Spring整合web项目原理



1. 实现原理
2. ServletContext对象
3. 监听器（监听它什么时候创建）
4. 具体使用

在服务器启动时候，为每个项目创建一个ServletContext对象

在ServletContext对象创建的时候，使用监听器可以知道ServletContext对象在什么时候创建

使用监听器听到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）

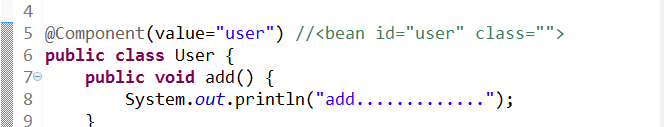
获取对象时候，到ServletContext域中得到（getAttribute方法）

# Spring的bean管理（注解方式





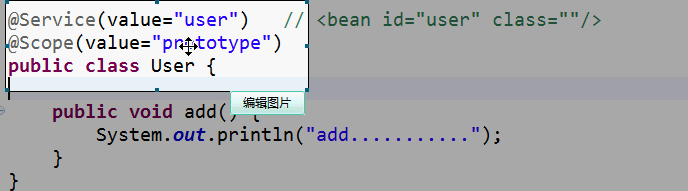
# 注解创建对象



1. 创建对象有4个注解
2. 
   1. @Component
   2. @Controller ：WEB层
   3. @Service ：业务层
   4. @Repository ：持久层

目前这四个注解功能是一样的，都是创建对象

1. 创建对象是单实例还是多实例







# 配置文件和注解混合使用

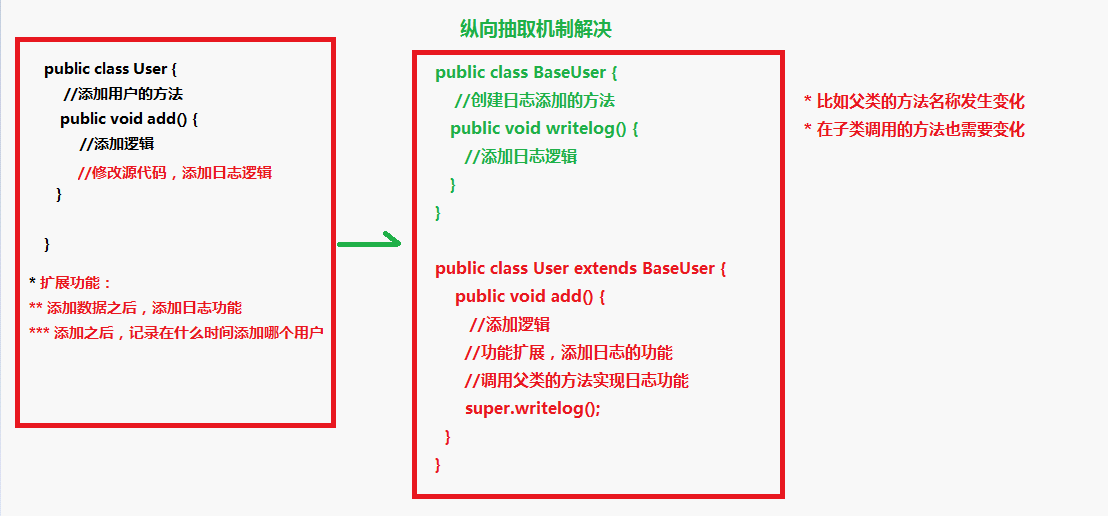
1创建文件操作使用配置文件方式实现

2注入属性操作使用注解方式实现

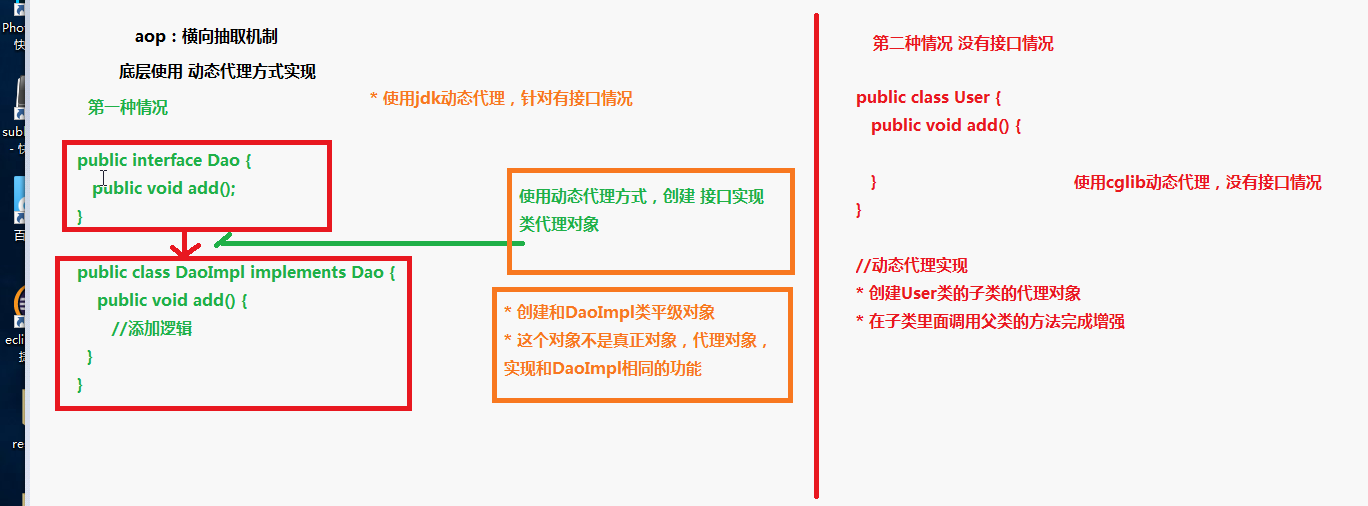
# AOP

1. aop概述
   1. aop：面向切面（方面）编程，扩展功能不修改源代码实现
   2. aop采取了横向抽取机制，取代了传统纵向继承体系重复性代码
2. aop底层原理

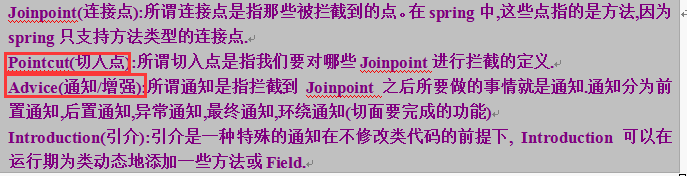
画图分析

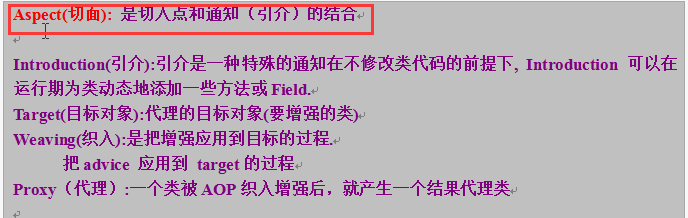






1. aop操作术语





JoinPoint（连接点）：类里面可以被增强的方法，这些方法称为连接点

PointCut（切入点）：在类中可以有很多的方法被增强，比如实际操作类中，只增强了类里面add方法和update方法，实际增强的方法称为切入点

Advice（通知/增强）：增强的逻辑，称为增强，比如扩展日志功能，这个日志功能称为增强

前置通知：在方法之前执行

后置通知：在方法之后执行

异常通知：方法出现异常

最终通知：后置之后来执行

环绕通知：在方法之前和之后执行

* 切面：把增强应用到具体方法上面，过程称为切面（把增强用到切入点过程）

# Spring的aop操作

1在spring里面进行aop操作，使用aspectj实现

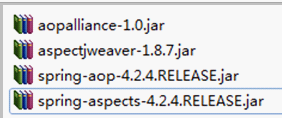
1. Aspectj不是spring一部分，和spring一起使用进行aop操作
2. Spring2.0以后增强了对Aspectj支持
3. 建议都使用它来实现aop操作

2使用aspectj实现aop有两种方式

1. 基于aspectj的xml配置
2. 基于aspectj的注解方式

# Aop操作准备

1. 除了导入最基本的jar包还需要导入与aop相关的jar包



1. 创建spring核心配置文件，导入aop约束

# 使用表达式配置切入点

1 切入点：实际增强的方法

2 常用的表达式



（1）execution(<\* zll.cwnu.aop.Book.add(..)>) \*访问修饰符

（2）execution(<\* zll.cwnu.aop.Book.\*(..)> Book类里面的所有方法

（3）execution(<\* \*.\*(..)> 所有类的所有方法

（4）execution(<\* save\*(..)> 以save开头的所有方法





# Log4j介绍

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

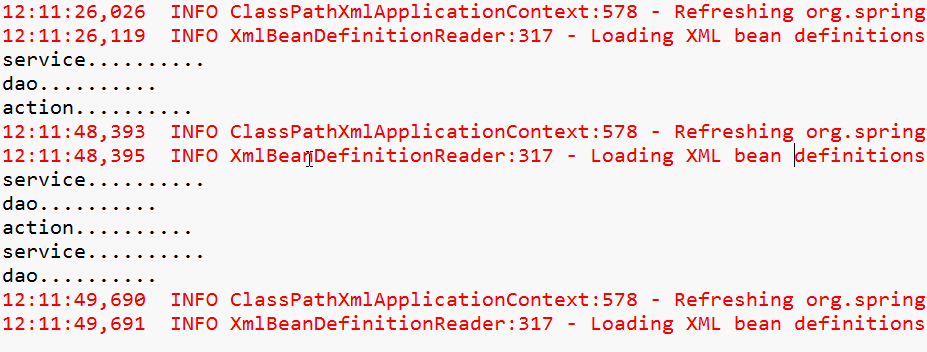
2 使用

1. 导入log4j的jar包
2. 复制log4j的配置文件，复制到src下

Log4j.properties

# Spring整合web项目演示

1. action调用service,service调用dao



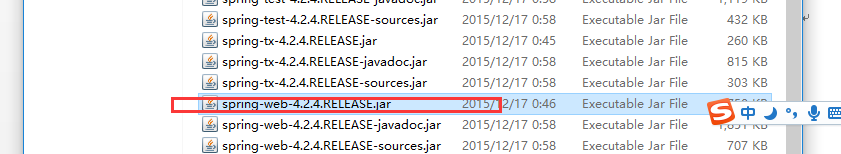
每次访问action时候，都会加载spring配置文件

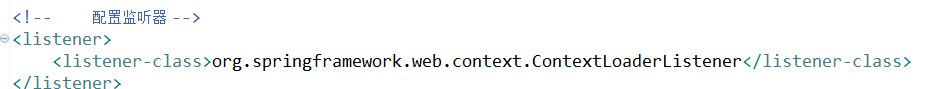
2 解决方案

1. 在服务器启动时候，创建对象加载配置文件
2. 底层使用监听器、ServletContext对象

3在spring里面不需要我们自己写代码实现，帮我们封装了

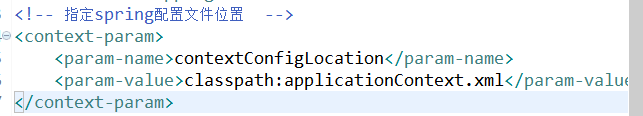
1. 封装了一个监听器，只需要配置监听器即可
2. 配置监听器做事情：导入spring整合web项目jar包





1. 指定加载spring配置文件的位置





# Sprig的jdbcTemplate操作

1 spring框架一站式框架

1. 针对javaee三层，每一层都有解决技术
2. 在dao层，使用jdbcTemplate

2 spring对不同的之久化层都进行了封装

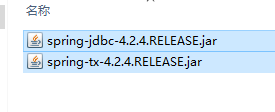


1. jdbcTemple对jdbc进行封装

3 jdbcTemplate使用和dbutils使用很相似，都是对数据库进行crud操作

添加

1导入jdbcTemple相关的jar包

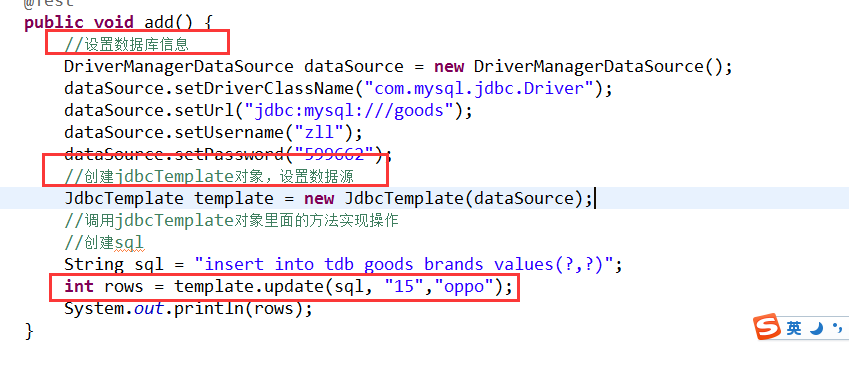


2 创建对象，设置数据库信息

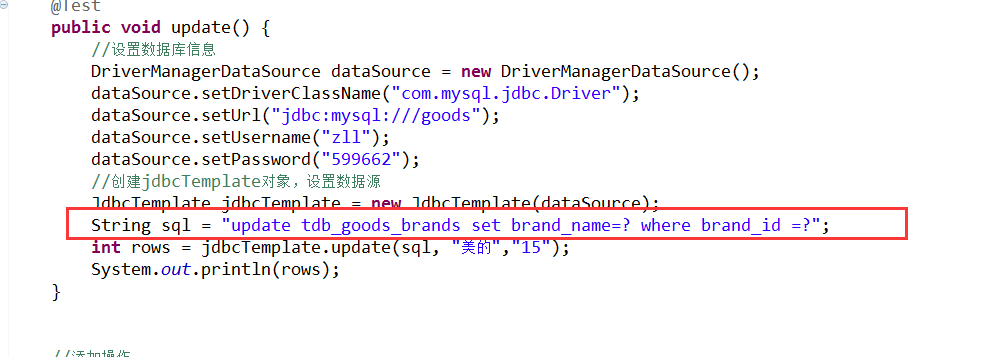
3 创建jdbcTemplate对象，设置数据源

4 调用jdbcTemplate对象里面的方法实现操作

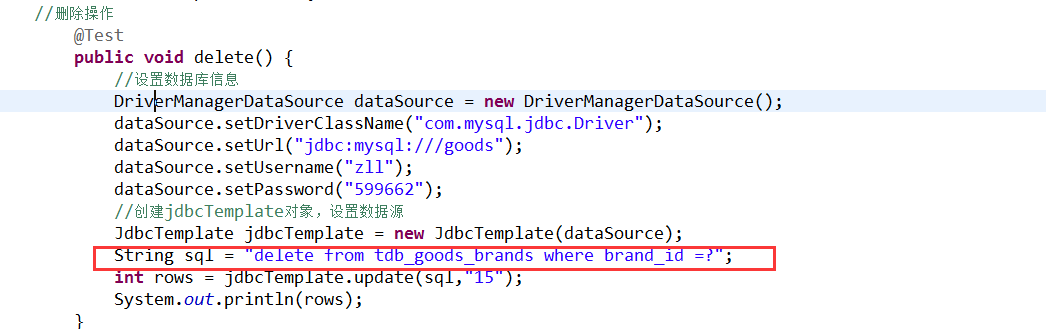




修改



删除



查询

第一个 查询返回某一个值

第二个 查询返回对象

第三个 查询返回list集合

# Spring配置连接池哈dao使用jdbcTemplate

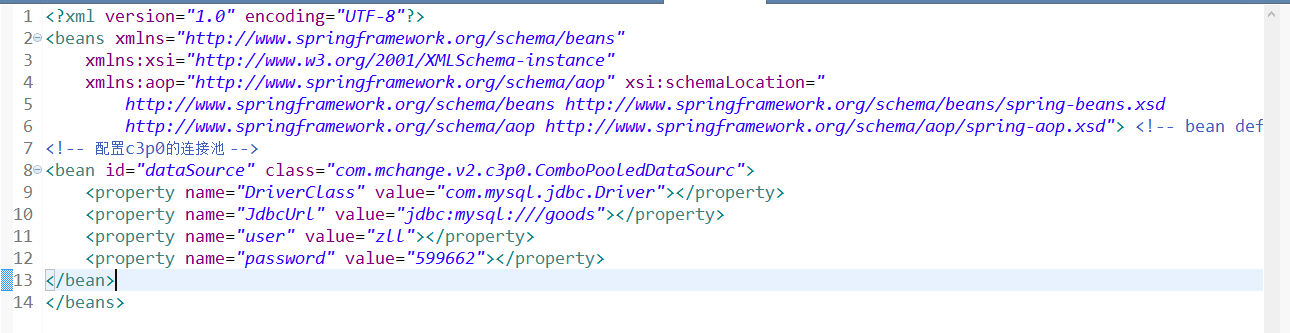
1. spring配置c3p0连接池

第一步：导入jar包



第二步：穿件spring配置文件







<!-- 配置c3p0的连接池 -->

<!-- 创建UserDao和UserService对象，完成service对象中注入dao对象 -->

<!-- 创建JdbcTemplate对象 -->

<!-- 把dataSource传到模板中去 -->

# Spring的事务管理

## 事务概念

1. 什么是事务
2. 事务特性
3. 不考虑隔离性产生读问题
4. 解决读问题

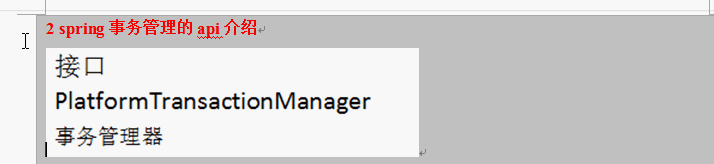
设置隔离级别

## Sping事务管理api

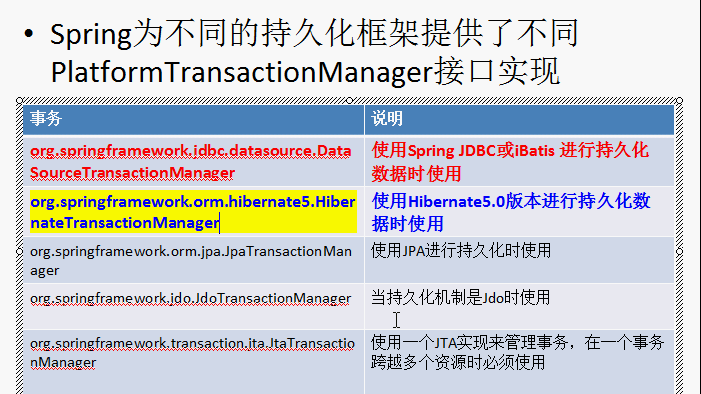
第一种 编程式事务管理（不用）

第二种 申明式事务管理

1. 基于xml配置文件实现
2. 基于注解实现



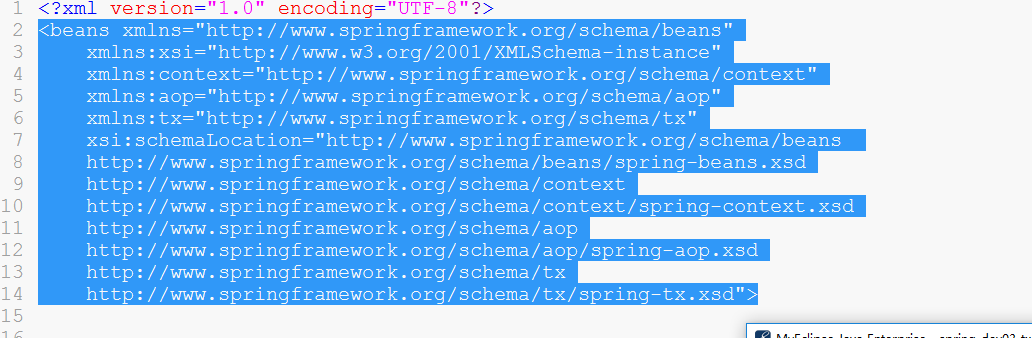
（1）Spring针对不同的dao层框架，提供接口不同的实现类



1. 首先配置事务管理器

### 搭建转账环境

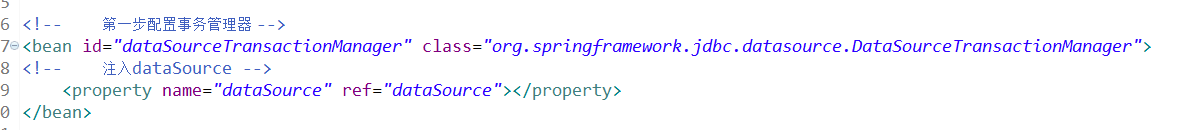
1. 创建数据表
2. 创建service和dao类，完成注入关系
3. service层又叫业务逻辑层
4. dao层，单纯对数据库操作层，在dao层不添加业务



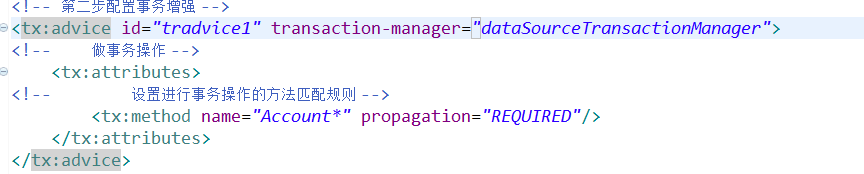
### 声明式事务管理（xml配置）

1. 配置文件使用aop思想配置

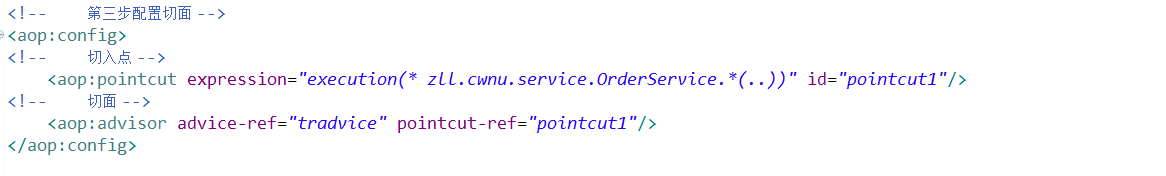
第一步 配置事务管理器



第二步 配置事务增强



第三步 配置切面



### 声明式事务管理（注解）

第一步 配置事务管理器

第二步 配置事务注解

第三步 在要使用事务的方法所在类上面添加注解



# SSH三大框架整合

1. struts2
2. hibernate5.x
3. spring4.x

# 回顾SSH框架知识点

## Hibernate框架

1 hibernate核心配置文件

1. orm思想

对象关系映射（实体类与数据库对应）

1. 数据库信息
2. Hibernate信息
3. 映射配置
4. Hibernate核心配置文件
   1. 如果单纯使用hibernate框架，核心配置文件名称hibernate.cfg.xml和位置src下面是固定的
   2. Hibernate和spring整合时候，hibernate核心配置文件名称和位置没有固定要求
5. hibernate映射配置文件
   1. 实体类和数据库表映射关系-------使用orm思想
6. hibernate操作的步骤
   1. 在spring框架对hibernate框架进行封装，使用hibernateTemplate

### Struts2框架

1. Action操作

（1）Action创建三种方式

继承类ActionSupport

（2）配置action访问路径

创建struts.xml配置文件。这个文件名称和位置固定src下面的

（3）配置访问action的多个方法

使用通配符方式配置

（4）在action获取表单提交数据

获取request对象

属性封装

模型驱动

（5）在action操作域对象

使用ServletActionContext获取域对象

1. 配置struts2过滤器
2. 值栈
   * 1. 向值栈中放数据

Set方法

Push方法

定义变量，生成get方法

* + 1. 从值栈中获取数据

在jsp中使用struts2标签+ognl获取

<s:property>

<s:iterator>

1. 拦截器
   * 1. aop和责任链
     2. 自定义拦截器

继承MethodFillterInterceptor

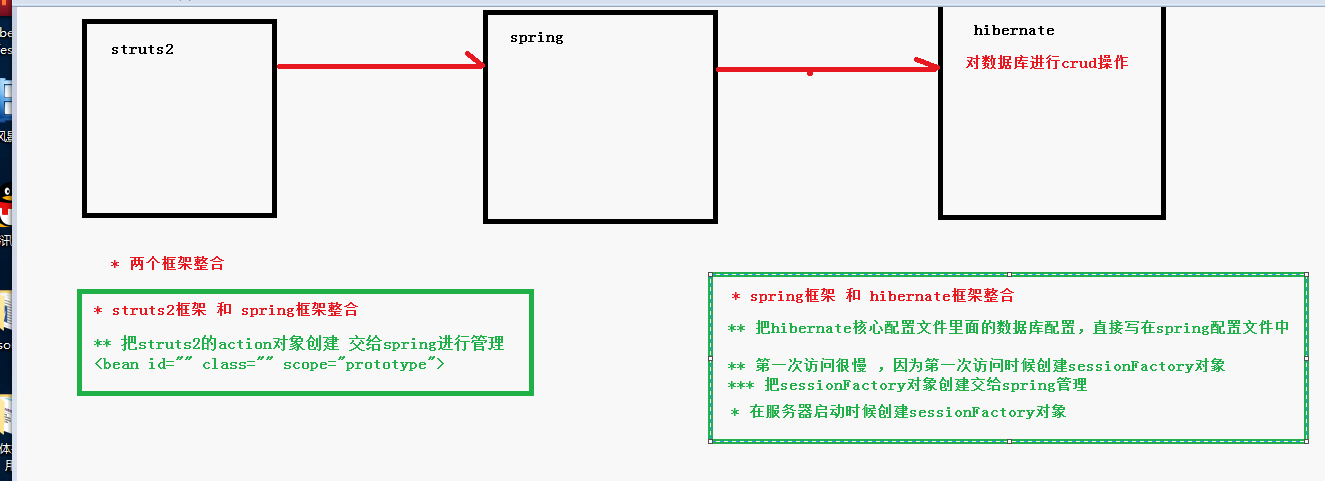
重写类里面的方法

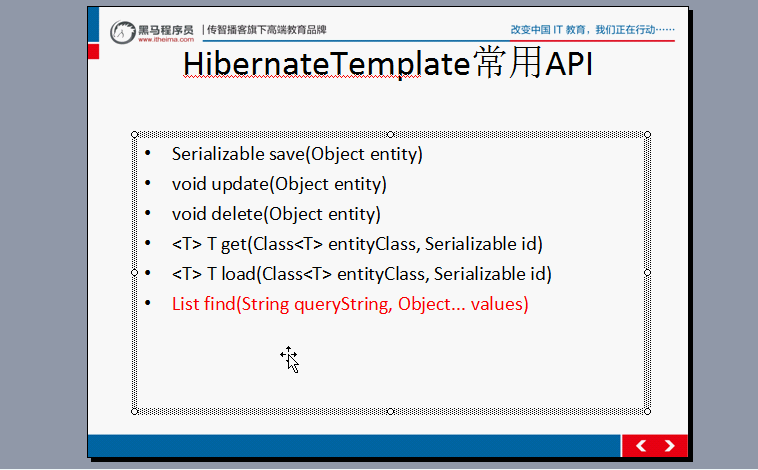
配置拦截器和action关联

### Spring框架

1. spring核心配置文件
2. 名称和位置没有固定要求
3. 在spring核心配置文件中引入schema约束
4. 创建对象
5. xml配置方式：<bean id=“” class=“”>
6. 注解方式：四个注解
7. 注入属性
8. xml配置方法
9. 注解方式:两个注解
10. 使用ServletContext对象和监听器实现
11. 在服务器启动时候，加载spring配置文件，创建对象
12. 配置spring的监听器
13. 指定spring配置文件位置
14. spring事务配置
15. xml方式
16. 注解方式

# SSH框架整合思想



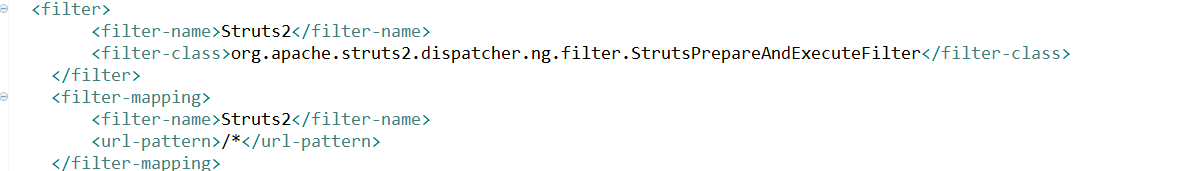


SSH框架整合过程

第一步 导入jar包

第二步 搭建strut2环境

1. 创建action，创建struts.xml配置文件，配置action
2. 配置struts2的过滤器



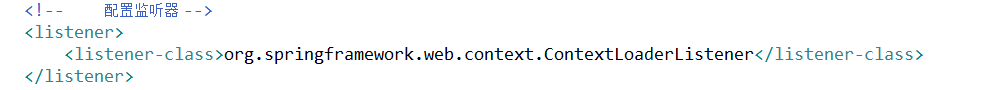
第三步 搭建hibernate环境

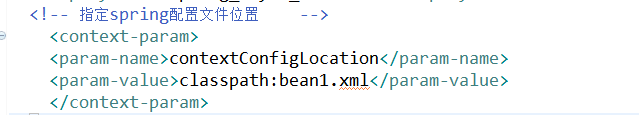
1. 创建实体类
2. 配置实体类和数据表映射关系
3. 创建hibernate核心配置文件

引入映射配置文件

第四步 搭建spring环境

1. 创建spring核心配置文件
2. 让spring配置文件在服务器启动时候加载



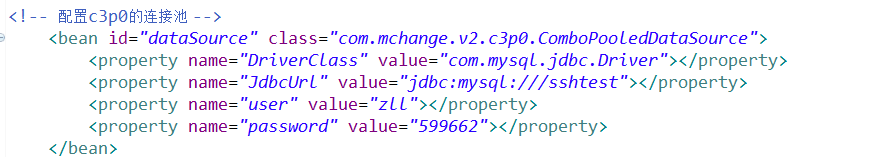


第五步 struts2和spring整合

1. 把action在spring配置（action多实例）
2. 在struts.xml中action标签class属性写bean的id值

第六步 spring和hibernate整合

1. 把hibernate核心配置文件中数据库的配置，在spring里面配置

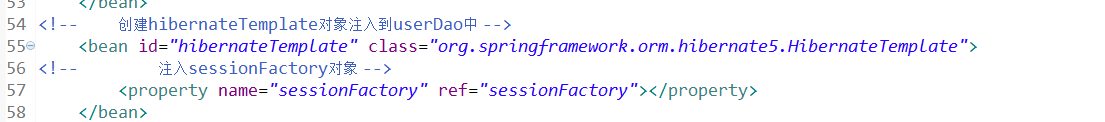


1. 把hibernate的sessionFactory在spring配置

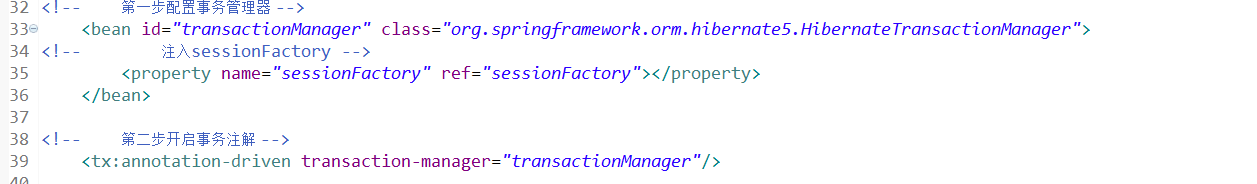


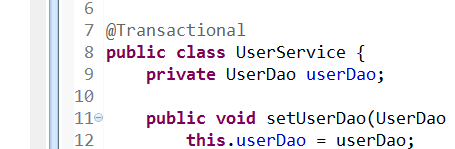
第七部 在dao里面使用hibernateTemplate

1. 在dao里面注入hibernateTemplate对象
2. 在hibernateTemplate对象中注入sessionFactory



第八步 配置事务

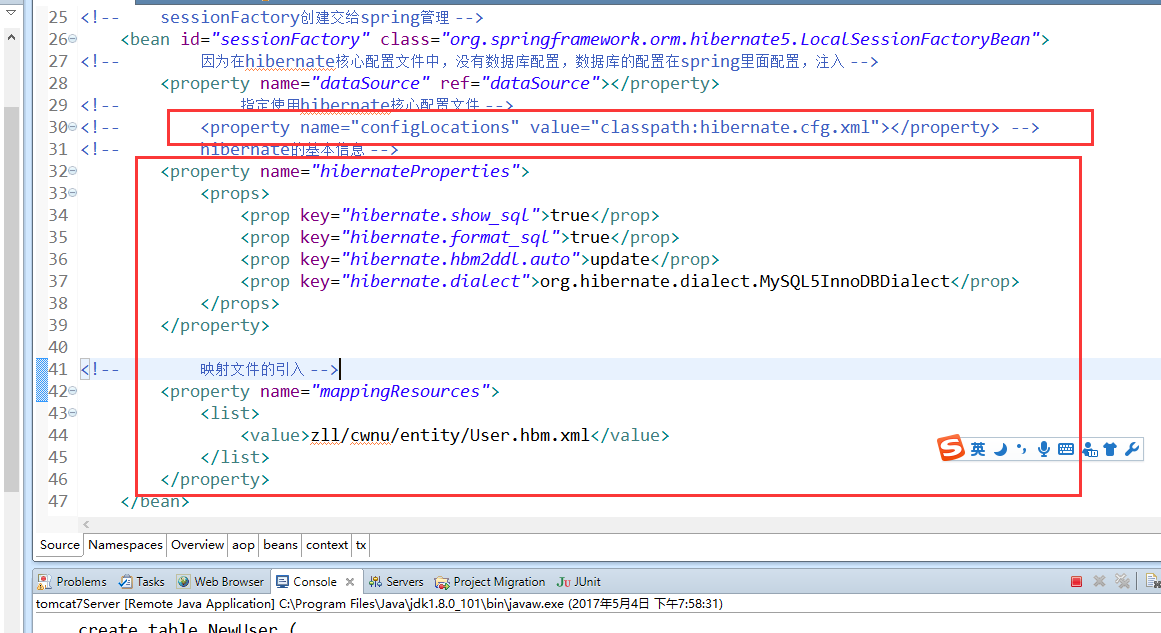




# 整合其他方式

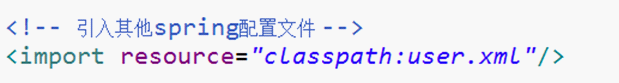
1. spring整合hibernate时候，可以不写hibernate核心配置文件

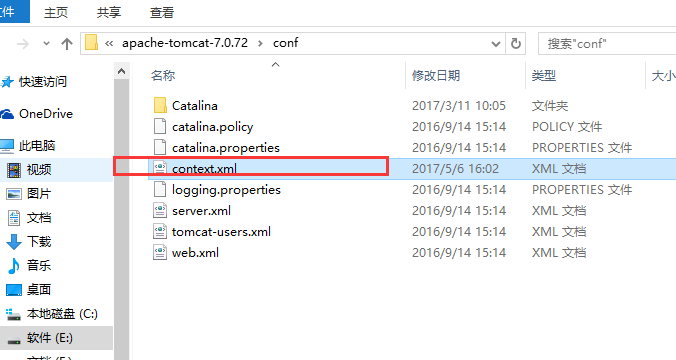
（1）把hibernate核心配置文件，基本信息配置和映射都放到spring中

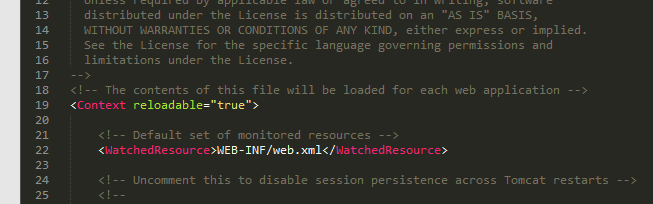


# Spring分模块开发

1. 在spring里面配置多个内容，造成配置混乱，不利于维护
2. 把spring核心配置文件中，一部分配置放到单独配置文件中，在spring核心配置文件中引入单独配置文件





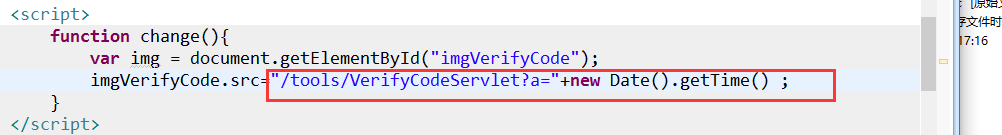


自动加载

# 图书管理项目

注意：在进行注册的时候，为了防止恶意注册，需要加上一个验证码

点击【看不清，换一张的时候】注意加上时间



避免不同浏览器的缓存，地址相同无法实现换一张的效果，每次的参数不同，才能干掉浏览器的缓存

注意：在进行表单的传递参数的时候，注意要用一个隐藏域（不要直接在后面加参数）

