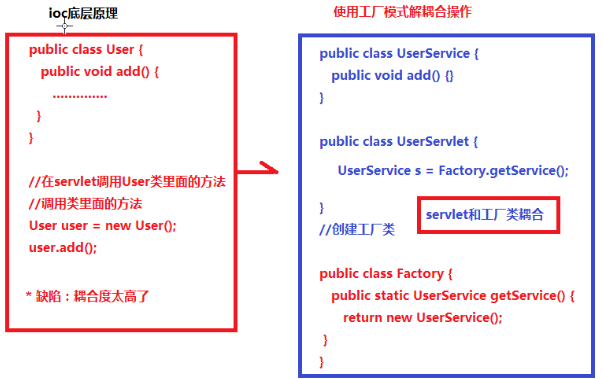
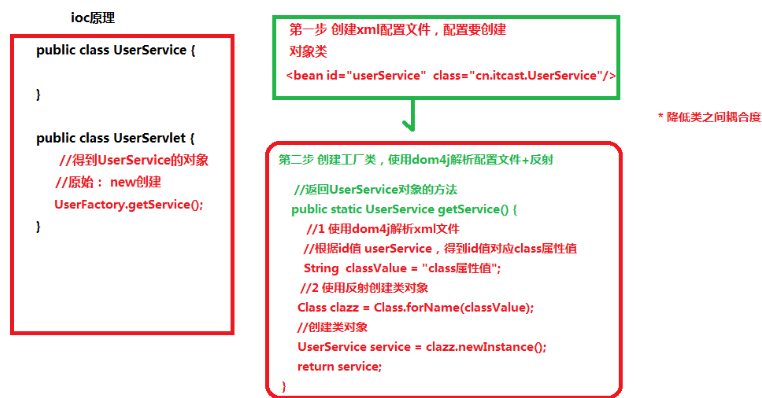
**Spring 学习**

1. spring\_day01
   1. Spring概念
      1. spring是开源的轻量级框架
      2. spring核心主要有两部分
         1. aop：面向切面编程，扩展功能不是修改源代码实现
         2. IoC：控制反转，比如有个类，类里面有方法（不是静态方法），调用类的方法，使用对象调用方法，创建类对象的过程，需要new出来对象，而spring把对象的创建不是通过new形式实现，而是交给spring配置创建类对象。
         3. spring是一站式框架
            1. spring在javaee三层结构中，每一层都提供不同的解决技术

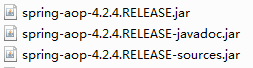
web层：springMVC，

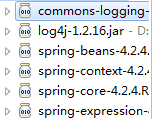
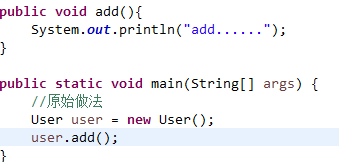
service层：spring的IoC

dao层：spring的jdbcTemplate

* + 1. spring版本
       1. spring4.x
  1. spring里面的IoC操作
     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
     3. IoC的底层原理
        1. IoC底层原理实用技术
           1. xml配置文件
           2. dom4j解决xml
           3. 工厂设计模式
           4. 反射
     4. IoC的演变过程
        1. 
     5. IoC的底层原理
        1. 
     6. IoC入门案例
        1. 导入jar包
           1. 解压zip文件

特点：jar包，文档，源码

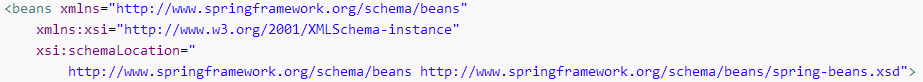


* + - * 1. 
        2. beans，，core，context，，expression
        3. 导入支持日志支持jar包
        4. 
      1. 创建类，在类里面创建方法
         1. 
      2. 创建spring配置文件，配置创建类
         1. spring核心配置文件的名称和位置不是固定的

建议放到src下面，官方建立applicationContext.xml

引入schema约束

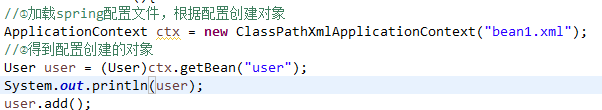
打开最后一个



配置对象的创建

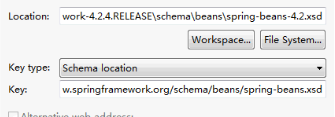


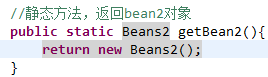
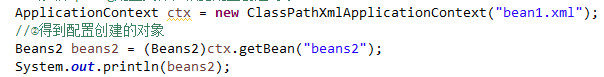
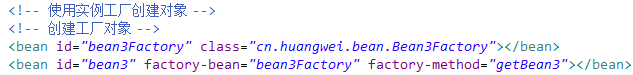
* + - 1. 写代码测试对象创建
         1. 这段代码中在测试中使用

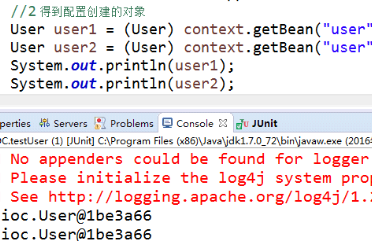


* + 1. 配置文件没有提示
       1. spring引入的schema约束，把约束文件引入到eclipse中
          1. 复试约束的路径

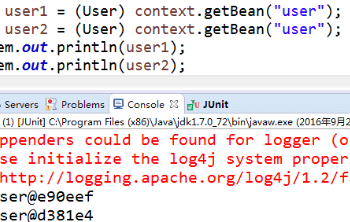




* 1. spring的bean管理（xml方式）
     1. 在spring里面通国配置文件创建对象
     2. bean实例化三种方式
        1. 使用类里面的无参数构造器创建（重点）
           1. 如果创建有参构造器，就不会自动调用无参构造器
           2. 
        2. 使用静态工厂创建
           1. 创建静态方法，返回类对象
           2. 
           3. 
           4. 
        3. 使用实例工厂创建
           1. 创建不是静态的方法，返回类对象
           2. 
     3. Bean标签的常用属性
        1. id属性：起名称，属性值名称任意命名
           1. 不能写中文，和特殊符号
           2. 根据id值得到配置对象
        2. class属性：创建对象所在类的全路径
        3. name属性：和id属性一样也能得到对象
        4. scope属性：
           1. singleton：默认值，单例

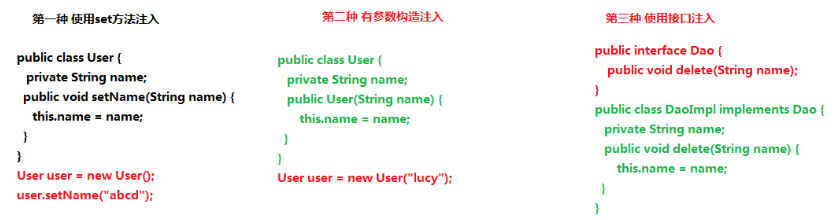
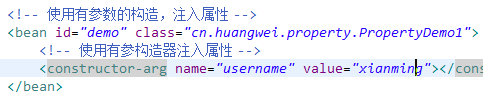
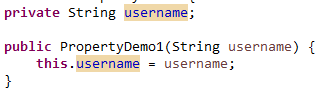
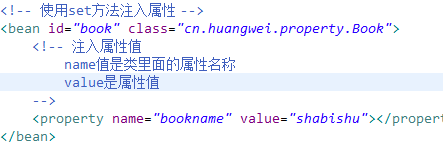
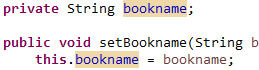


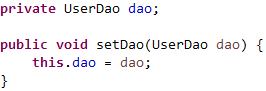
* + - * 1. prototype：多例，每次调用都会创建新的对象

* + - * 1. request：创建对象，把对象放到request域里面
        2. session：创建对象，把对象放到session域里面
        3. globalsession：创建对象，把对象放到globalsession域里面

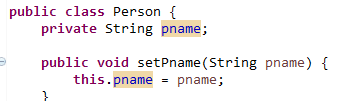
贴吧里面登陆过之后，其他文库都不用登陆

* 1. 属性注入
     1. 创建对象的时候，向类里面属性里面设置值
     2. 属性注入的方式介绍（三种方式）
        1. 
     3. spring框架里面只支持前两种方式
        1. set方式（重点）
        2. 有参数的构造器
     4. 有参数的构造注入属性
        1. 
        2. 
     5. 使用set方法注入属性
        1. 
        2. 
     6. 注入对象类型的属性（重点）
        1. 创建service类和dao类
           1. 在service得到dao对象
        2. 具体实现
           1. 在service里面把到作为类型属性
           2. 生成dao 类属性的set方法



* + - * 1. 配置文件中注入关系



* + 1. P名称空间的注入
       1. 
       2. 
       3. 
    2. 注入复杂类型的属性
       1. 数组类型
       2. list集合
       3. map集合
       4. properties类型

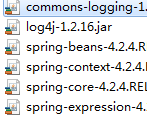
|  |
| --- |
| <!-- 注入复杂类型的属性值 -->  <bean id=*"person"* class=*"cn.huangwei.property.Person"*>  <!-- 数组 -->  <property name=*"arrs"*>  <list>  <value>往</value>  <value>吗</value>  <value>送</value>  <value>李</value>  </list>  </property>  <!-- list -->  <property name=*"list"*>  <list>  <value>小明</value>  <value>小红</value>  <value>小兰</value>  </list>  </property>  <!-- map -->  <property name=*"map"*>  <map>  <entry key=*"a"* value=*"lu"*></entry>  <entry key=*"b"* value=*"ma"*></entry>  <entry key=*"c"* value=*"to"*></entry>  <entry key=*"d"* value=*"je"*></entry>  </map>  </property>  <!-- properties -->  <property name=*"properties"*>  <props>  <prop key=*"driverclass"*>com.mysql.jdbc.Driver</prop>  <prop key=*"username"*>root</prop>  <prop key=*"password"*>123456</prop>  </props>  </property>  </bean> |

* 1. IoC和DI的区别
     1. IoC：控制反转个，吧对象创建交给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的基础上才能实现
  2. spring整合web项目原理
     1. 加载spring的核心配置文件



* + - 1. new对象，功能可以实现，但是效率很低
    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. 具体使用
          1. 在服务器启动时候，为每个项目创建ServletContext对象
          2. 在ServletContext对象创建时候，使用监听器可以监听到ServletContext对象在什么时候创建
          3. 使用监听器监听到ServletContext对象创建的时候，加载spring配置文件，把配置文件配置对象创建出来
          4. 把创建出来的对象放到ServletContext域对象里面（setAttribute方法）
          5. 获取对象的时候，到servletContext域对象得到（getAttribute方法 ）

1. spring\_day02
   1. Spring的bean的管理（注解）
      1. 注解
         1. 代码里面的特殊标记，使用注解可以完成相关功能
         2. 注解写法 @注解名称（属性名称=属性值）
         3. 注解可以使用在类上面，方法上面，属性上面
      2. spring注解开发准备
         1. 导入jar包
            1. 导入spring基本jar包



* + - * 1. 导入aop的jar包



* + - 1. 创建类和方法
      2. 创建spring配置文件，引入约束
         1. 第一天做ioc操作，引入beans约束
         2. 做spring的ioc注解开发，引入新的约束

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

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

xmlns:context="http://www.springframework.org/schema/context"

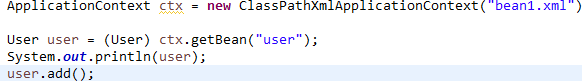
xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd

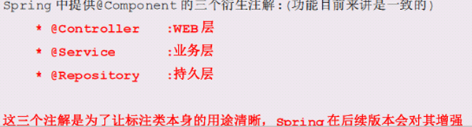
http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.2.xsd">

</beans>

* + - 1. 
    1. 注解创建对象
       1. 在创建对象的类上面使用注解实现



* + - 1. 
      2. 创建对象有四个注解

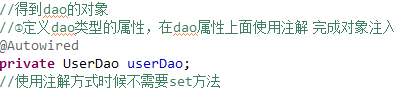


* + - * 1. @Component
        2. @Controller
        3. @Service
        4. @Repository
        5. 这四个注解的功能是一样的，都用来创建对象
      1. 创建对象是单实例还是多实例
         1. 
    1. 注解注入属性
       1. 创建service类，创建dao类，在service中得到dao对象，注入属性第一个注解@Autowired
          1. 创建dao和service对象





* + - * 1. 在service类里面定义dao类型属性



使用autowired注解时，属性名和bean的id不一致也没问题，因为他会扫描里面的有component注解的类注入，但是一般名字要匹配

* + - * 1. 注入属性的第二个注解,@Resource



* + - * 1. 如果value值与resource里面的name值不一样



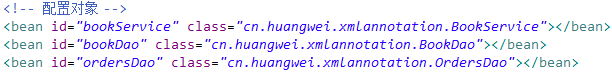
就会出现异常



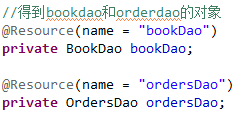
* + 1. 配置文件和注解混合使用

，很容易搞错

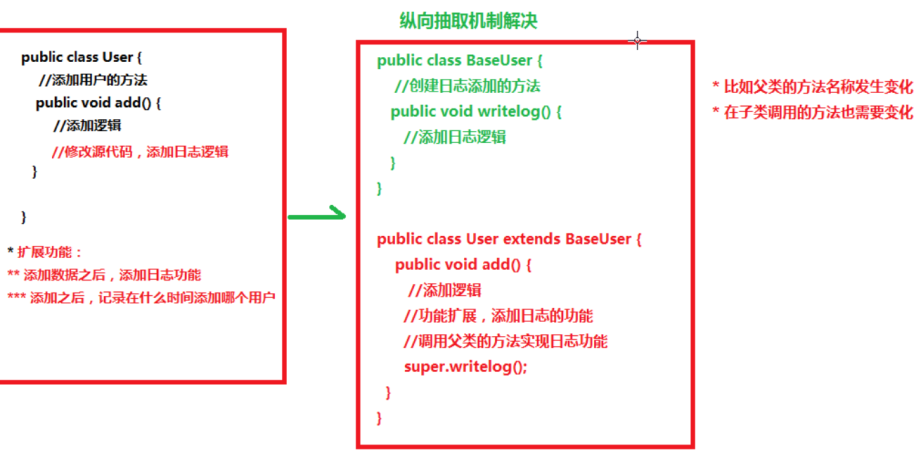
* + - 1. 创建对象操作使用配置文件方式实现



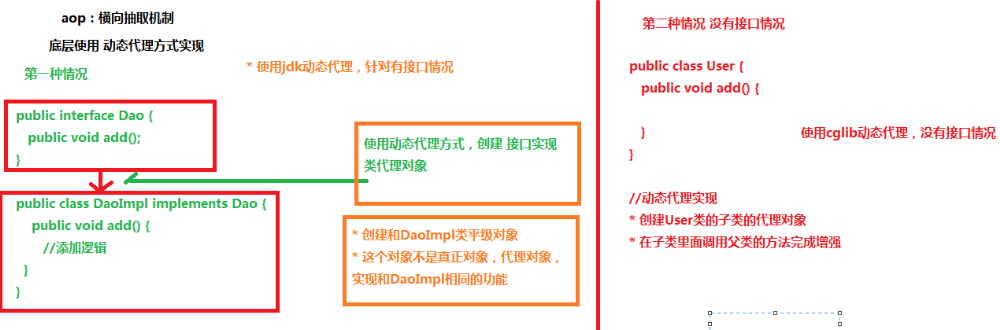
* + - 1. 注入属性的操作使用注解方式实现

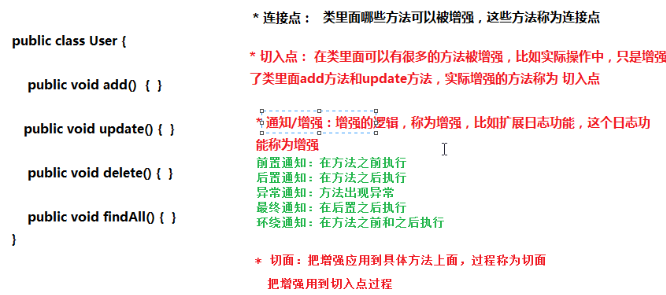


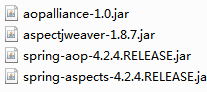
* 1. AOP
     1. aop概念
        1. 面向切面编程，扩展一个功能，不修改源代码，通过配置实现
        2. aop采取横向抽取机制，取代了传统纵向继承体系重复性代码
     2. aop原理
        1. 变化过程---纵向抽取机制

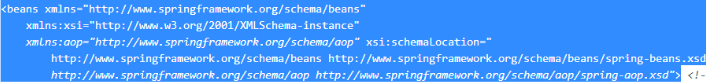


* + - 1. 横向抽取机制



* + 1. aop操作术语
       1. JoinPoint(连接点)：类里面的哪些方法可以被增强的方法
       2. 
       3. PointCut:在类里面可以有很多方法被增强，实际操作中，只增强add方法和update方法，实际被增强的方法就是切入点
       4. Advice（通知、增强）：增强的逻辑，成为增强，比如扩展日志的功能，这个日志的功能的逻辑就是增强
          1. 前置通知：在方法之前执行
          2. 后置通知：在方法之后执行
          3. 异常通知：方法中出现异常的通知
          4. 最终通知：后置通知之后执行
          5. 环绕通知：方法之前和之后执行
       5. Aspect（切面）：把增强具体应用到你的方法上面，过程叫切面，把增强用到切入点的过程
       6. Target：增强方法所在的类
       7. Weaving（织入）：把增强用到类的过程
       8. proxy代理：
  1. Spring里面aop的使用
     1. 在spring里面进行aop的操作，使用AspectJ实现
        1. aspectJ不是spring的一部分，和spring一起使用进行aop操作
        2. spring2.0以后新增了对aspectJ的支持
     2. 使用aspectJ实现aop的两种方式
        1. 基于aspectJ 的xml的配置
        2. 基于aspectJ的注解方式
     3. Aop的操作准备
        1. 除了导入最基本的jar包之外还需要导入aop的相关jar包



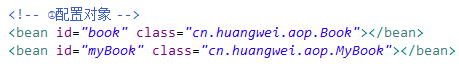
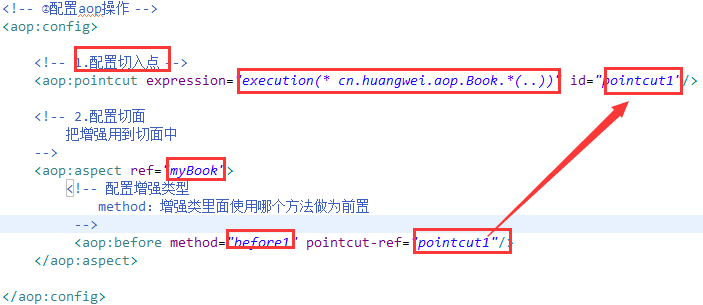
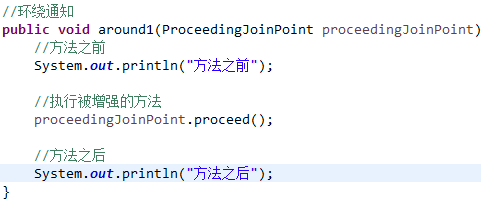
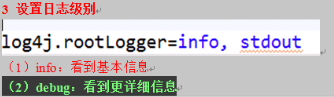
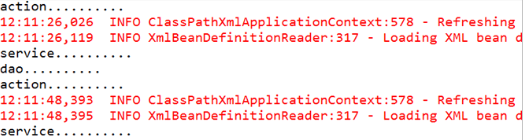
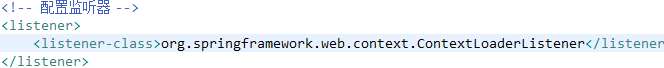
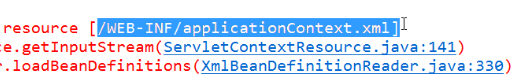
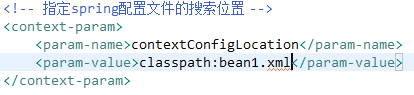
* + - 1. 创建spring核心的配置文件，导入aop的约束
         1. 
      2. 使用表达式配置切入点
         1. 切入点：实际要增强的方法
         2. 常用的表达式execution(<访问修饰符>?<返回类型><方法名>(<参数>)<异常>)

execution(\* cn.huagnwei.aop.Book.add(..))

execution(\* cn.huagnwei.aop.Book.\*(..))

execution(\* \*.\*(..))

匹配所有save开头的方法execution(\* save\*(..))

* + - 1. AspectJ的aop操作
         1. 
         2. 
         3. 
  1. log4j问题
     1. log4J介绍
        1. 通过log4j可以看到程序运行过程中更详细的信息
        2. 经常使用log4j查看日志
        3. 使用
           1. 导入log4j的jar包
           2. 复制log4j的配置文件，复制到src下面
        4. 
  2. spring整合web项目
     1. 
     2. 每次访问action的时候，都会加载spring配置文件，性能受影响
     3. 解决方案：
        1. 在服务器启动时候，创建对象加载配置文件
        2. 底层是用监听器，servletContext对象
     4. 在spring里面不需要我们自己写代码实现，已经帮我们封装好了
        1. 封装了一个监听器，只需要配置监听器就可以了
        2. 配置监听器之前，导入spring整合web项目的jar包
           1. 
           2. 
        3. 指定要加载的spring配置文件的位置
           1. 找不到
           2. 
           3. 

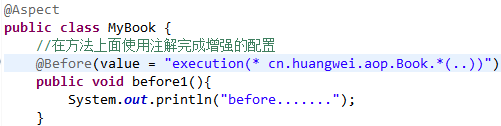
1. spring\_day03
   1. 基于aspectJ的注解aop
      1. 使用注解方式实现aop操作
         1. 创建类对象（配置文件）



* + - 1. 开启aop操作



* + - 1. 在增强类上使用注解完成增强的操作

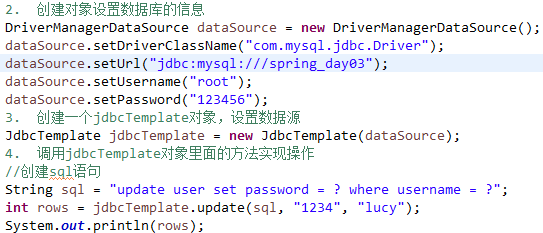


* 1. Spring的jdbcTemplate操作
     1. spring框架一站式框架
        1. 针对javaee三层，每一层都有解决方案
        2. 在dao层，使用jdbcTemplate
     2. spring对不同的持久化的技术进行了封装
        1. 
     3. jdbcTemplat使用和dbutils使用很相似，都对数据库进行crud操作
     4. 使用模板步骤：
        1. 导入jdbcTemplat的jar包

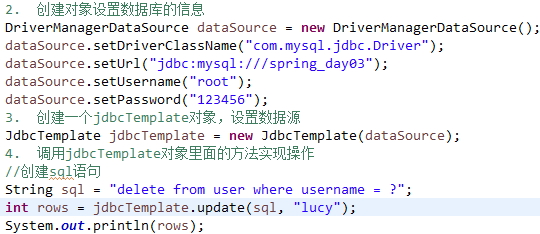




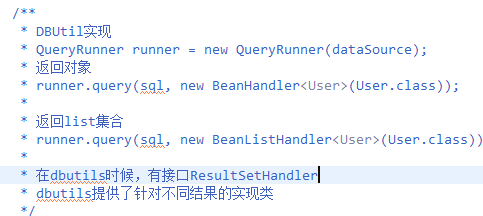
* + - 1. 创建对象设置数据库的信息
      2. 创建一个jdbcTemplate对象，设置数据源
      3. 调用jdbcTemplate对象里面的方法实现操作
    1. 实现操作
       1. 添加
          1. 
       2. 修改



* + - 1. 删除



* + - 1. 查询
         1. dbutils



* + - * 1. 使用jdbcTemplate实现查询操作

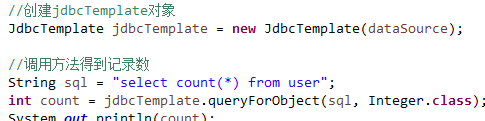
查询具体实现

查询返回某个值



第一个参数是sql语句

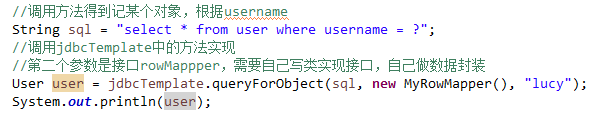
第二个参数是 返回类型的class ，String.class

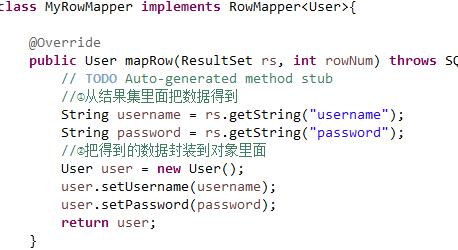


查询返回某个对象



sql，rowMapper（类似于resultSet）,可变参数

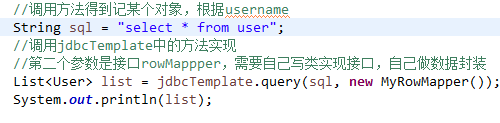


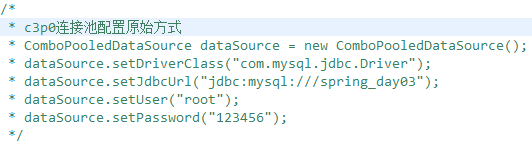
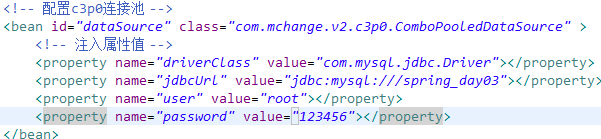


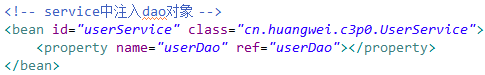
查询返回list集合



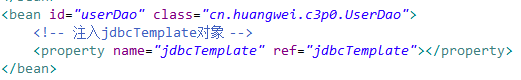
sql，rowmapper，自己写类实现封装，可变参数

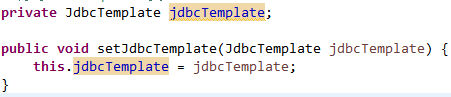


* 1. Spring配置连接池和dao使用jdbcTemplate
     1. spring配置c3p0连接池
        1. 导入jar包
           1. 
        2. 创建spring 配置文件，配置连接池
           1. 
           2. 
     2. dao使用jdbcTemplate
        1. 创建service和dao，配置service和dao对象，在service注入dao对象

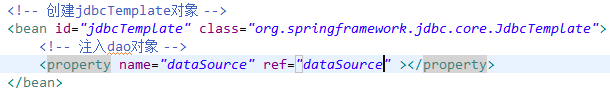


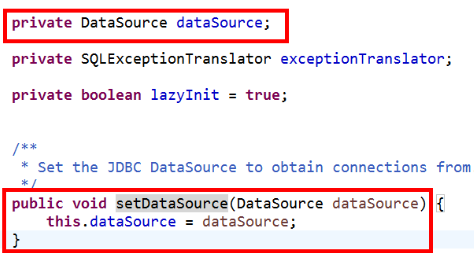
* + - 1. 创建jdbcTemplate对象，把模板对象注入到dao里面





* + - 1. 在jdbcTemplate里面注入dataSource





* + - 1. 