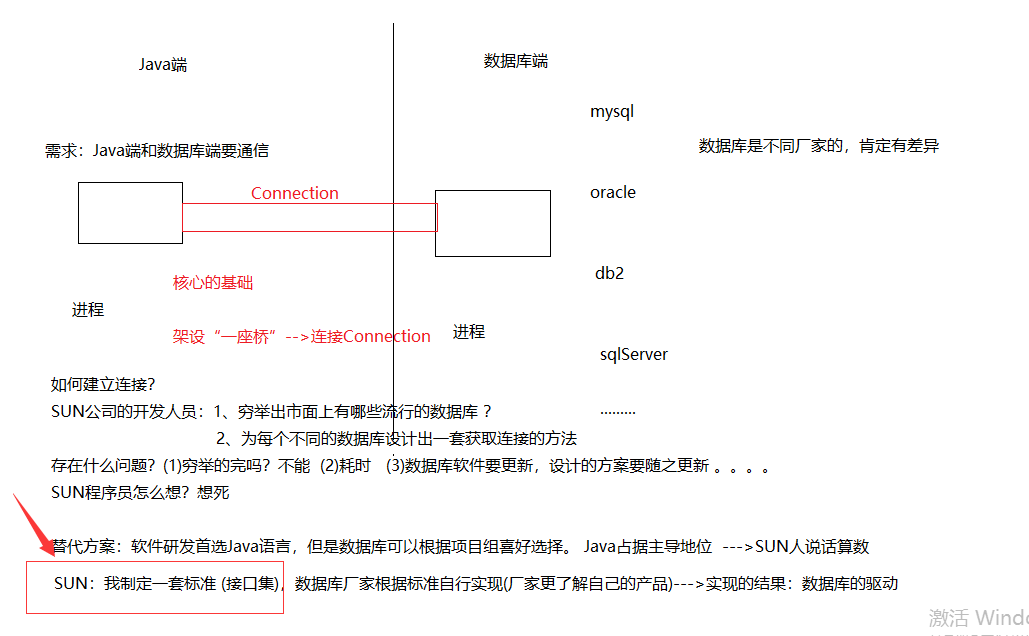
# 1.JDBC的简介

JDBC:java数据库连接技术

设计理念



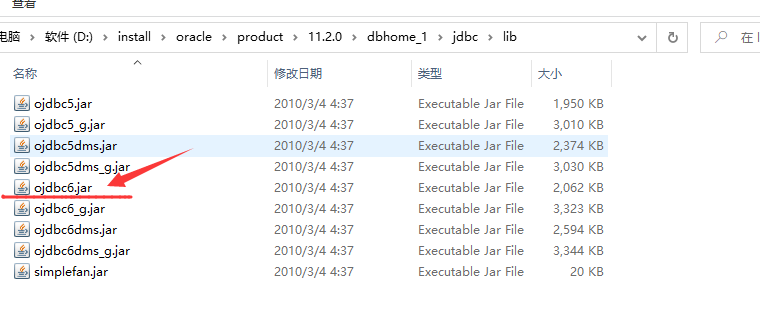
SUN负责设计接口标准

数据库厂家负责具体的实现---->驱动

包：java.sql下



oracle的驱动程序：

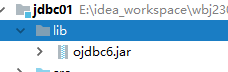


# 2.连通性测试

目标：能够让java程序和oracle数据库建立一个连接Connection

操作步骤：

(1)创建java工程，导入oracle的驱动包



(2)定义测试类



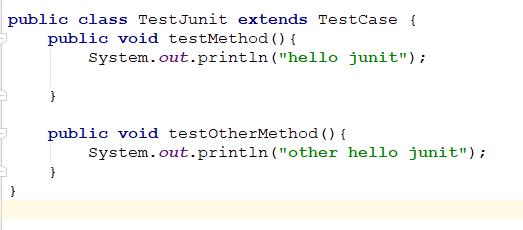
# 3.引入junit测试框架

(1)导入junit测试包

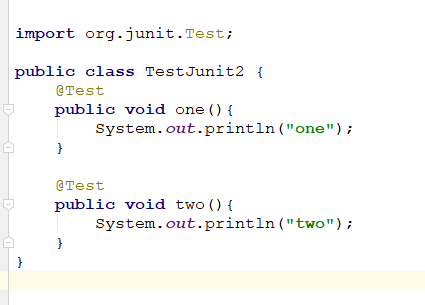
(2)使用

两种用法：

a.测试类需要继承TestCase,且方法必须是公共的，没有返回值，方法名以test打头的无参方法



b.普通类，且方法必须是公共的，没有返回值，的无参方法，使用@Test注解



# 4.CRUD操作

概念：实体对象

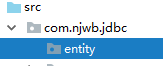
Java操作的是对象，数据库操作的是一个记录

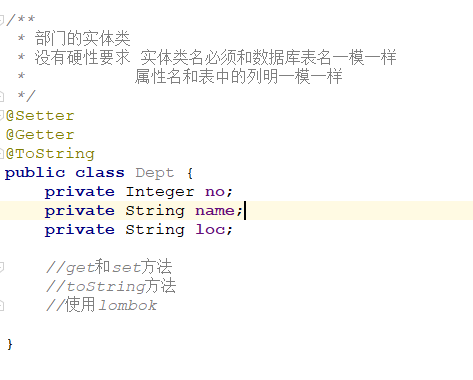


## 4.1 添加操作

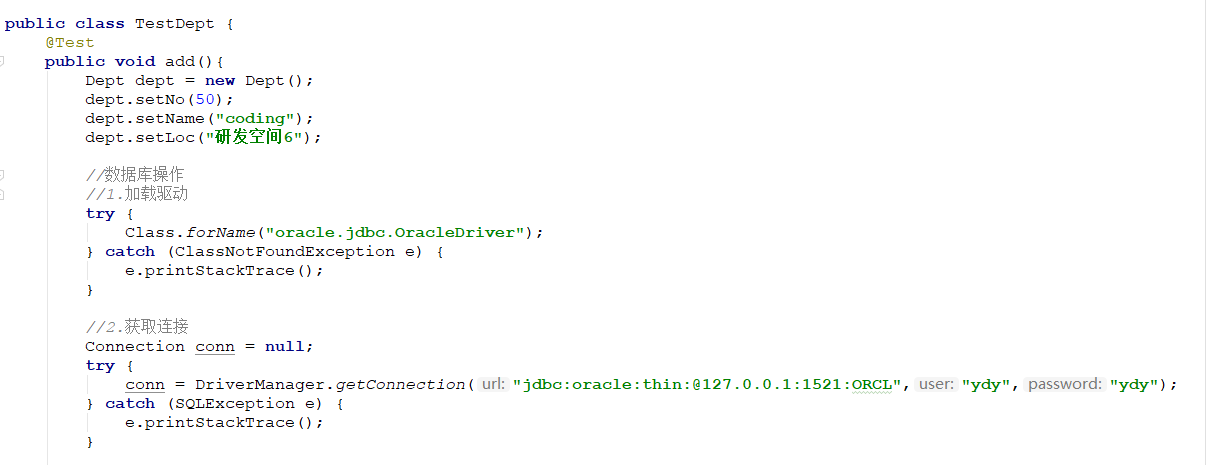
添加一个Dept对象

(1)定义一个实体对象Dept





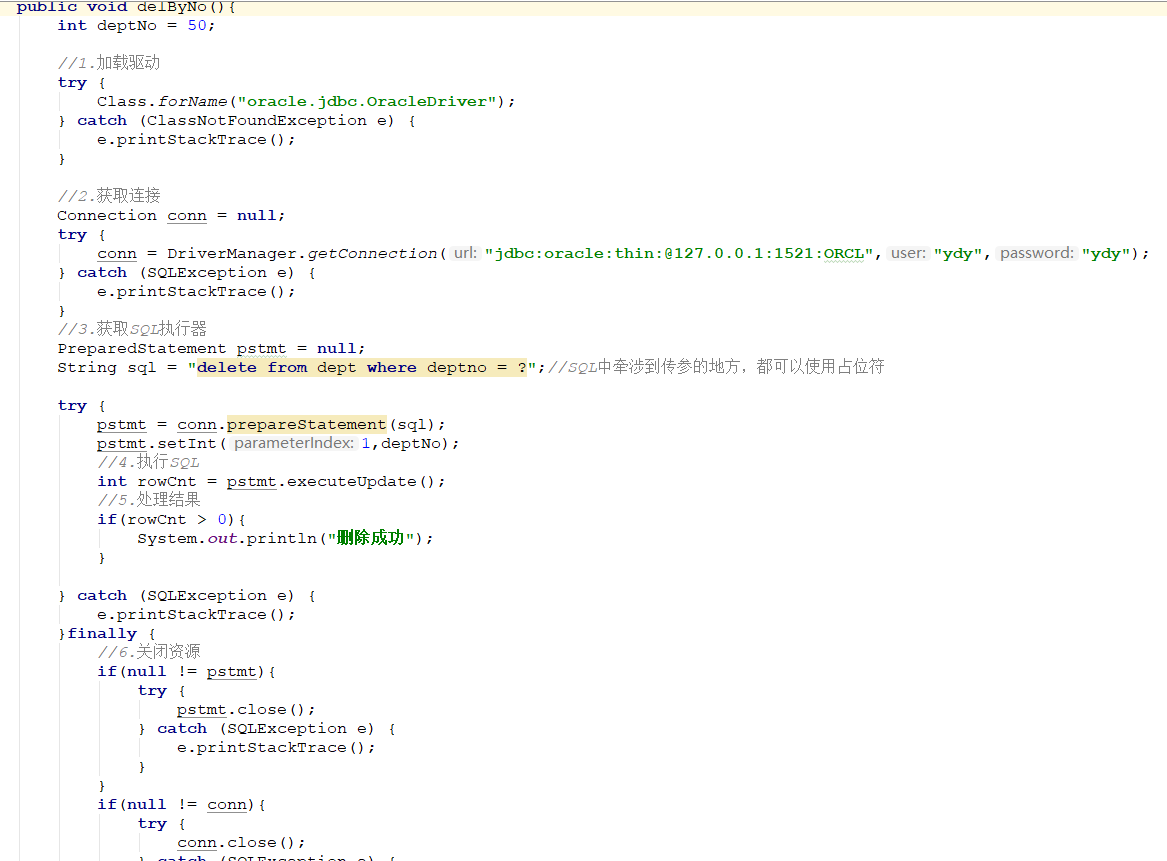
(2)往oracle数据库中添加一个Dept对象





## 4.2 删除操作

根据部门编号，删除一条dept数据



## 4.3 修改操作

修改指定的部门信息



## 4.4 初步封装JDBCUtil

分析过程：







使用



## 4.5 封装模板方法

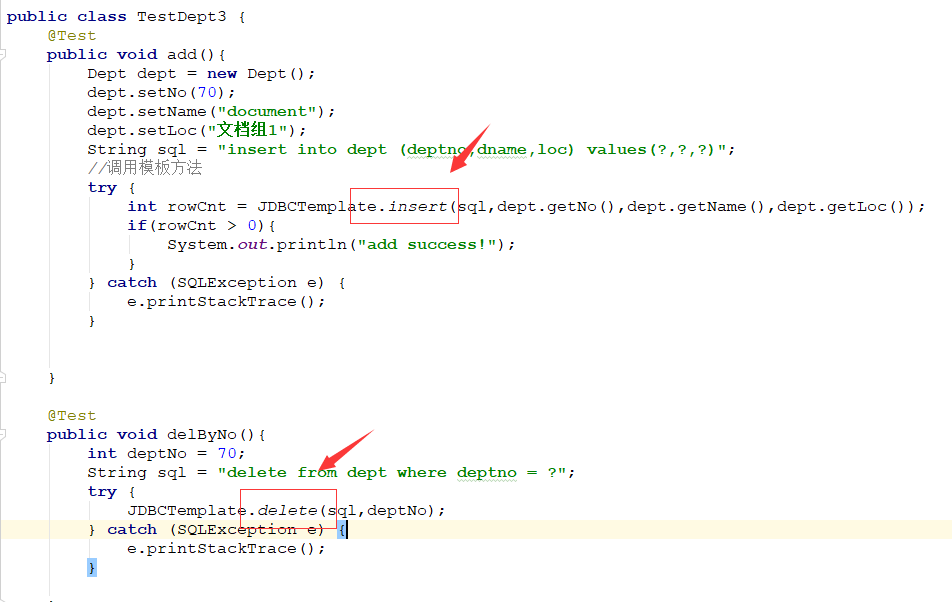


使用



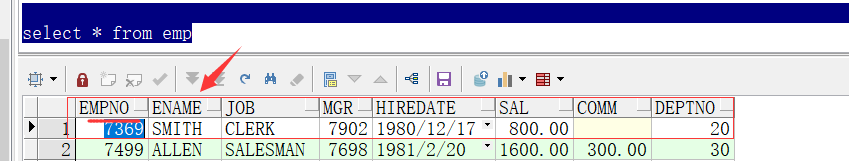
额外新增几个方法(语义的角度)



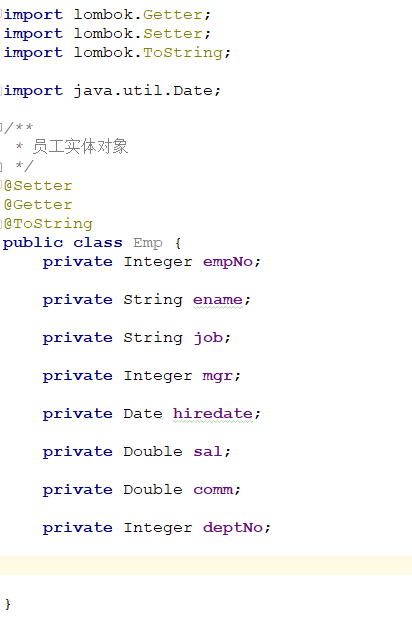


## 4.6 查询单个对象

需求：给定员工编号，查询员工信息

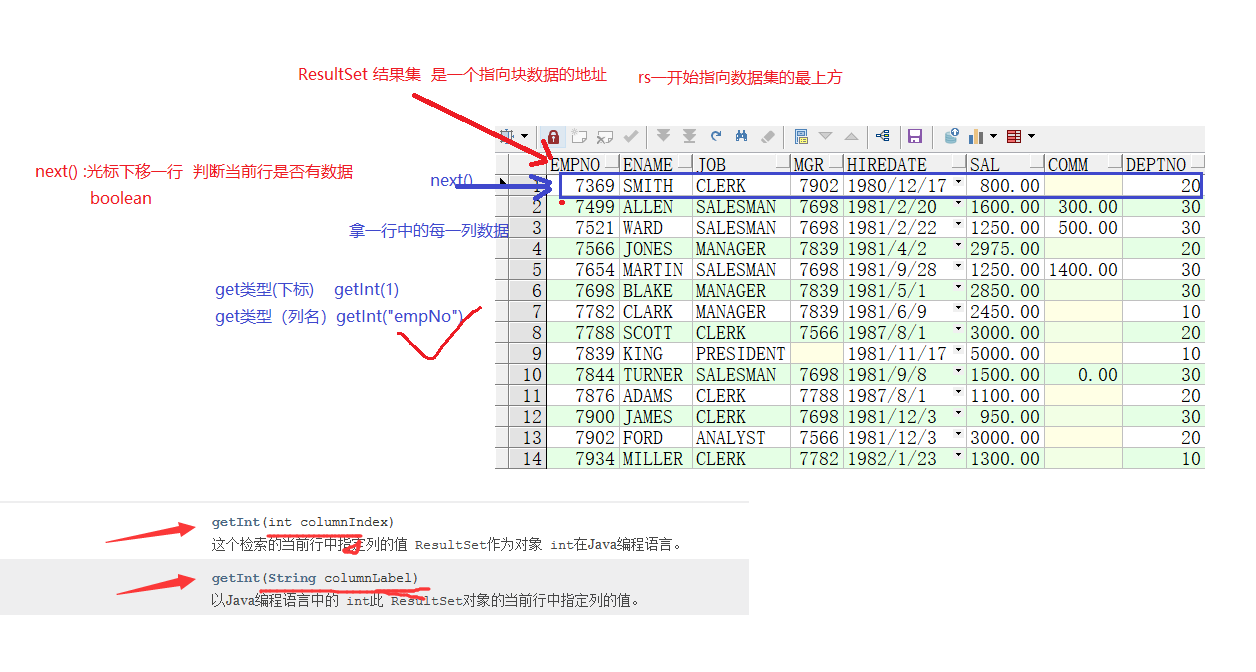


(1)实体类

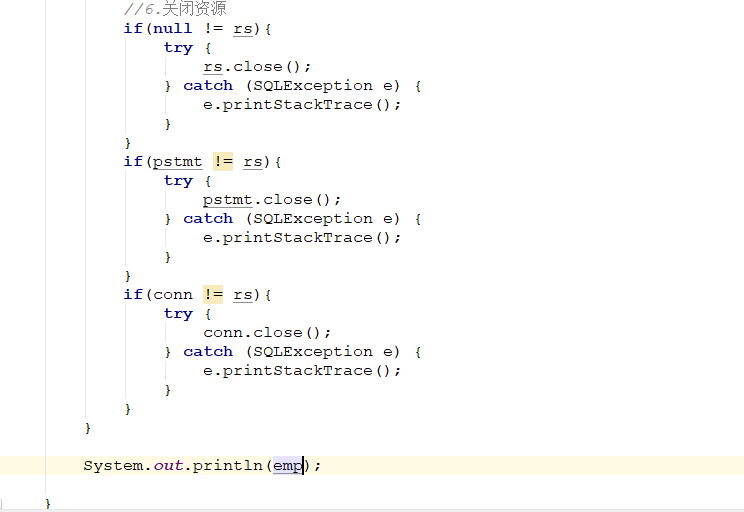


(2)给定员工编号，查询Emp对象

ResultSet的原理







## 4.7 查询多个对象



## 4.8 新增查询的模板方法





定义一个转换器接口



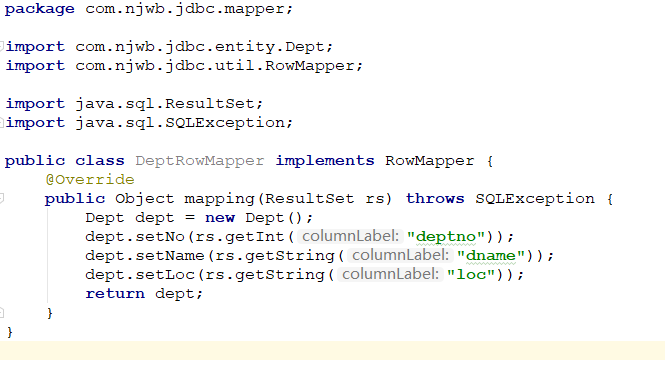
完成查询的模板方法

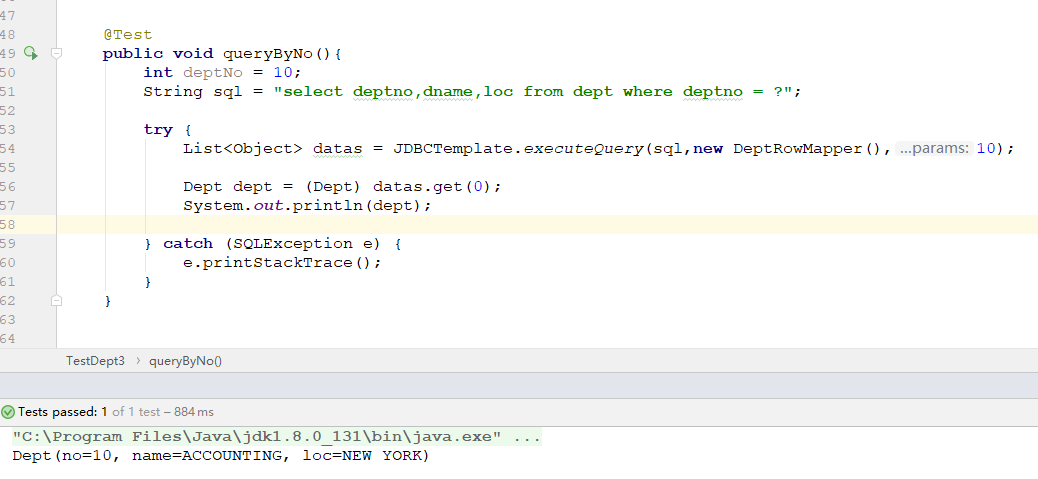


测试，查询单个对象

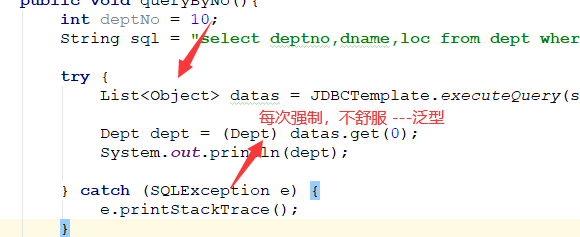
给定部门编号，查询部门信息





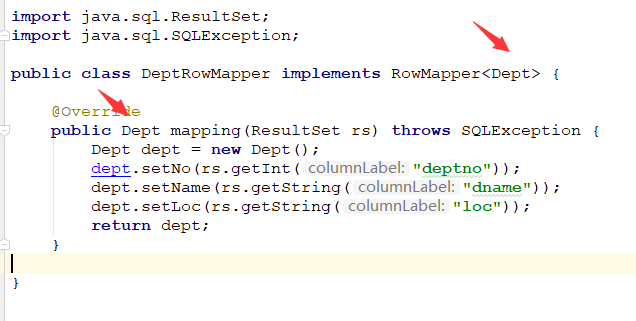


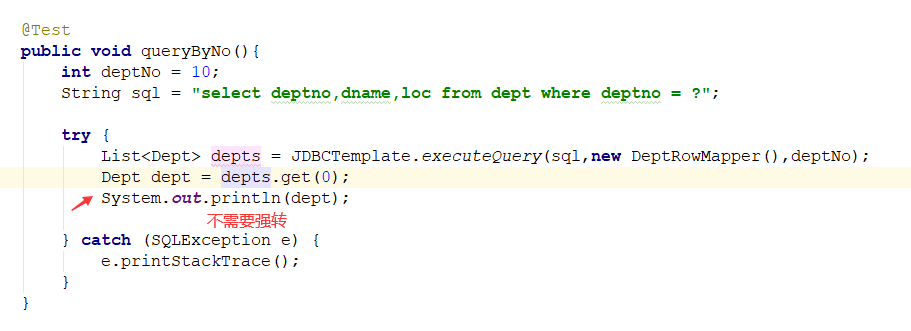
优化---泛型取代Object



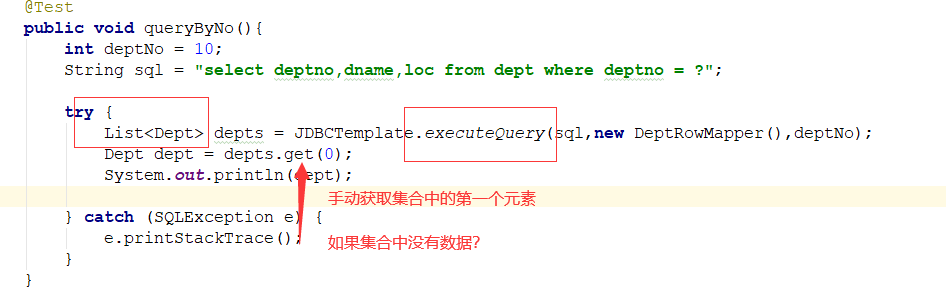


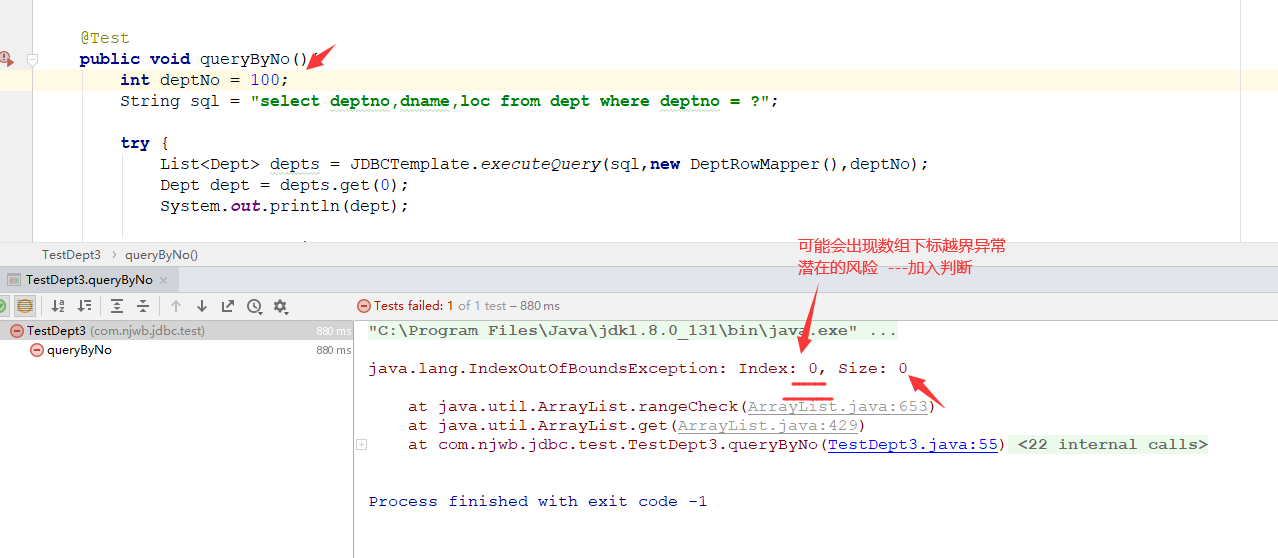




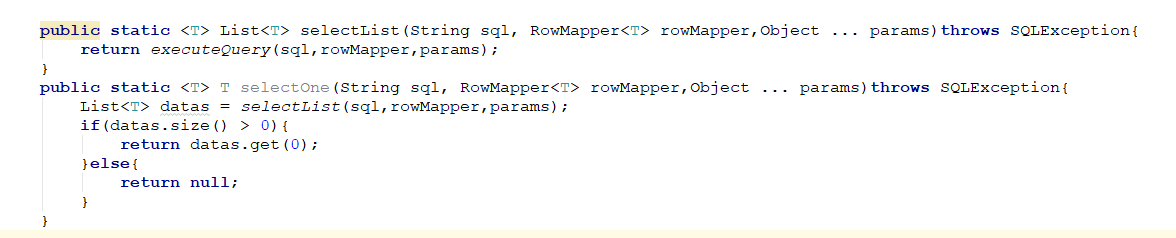


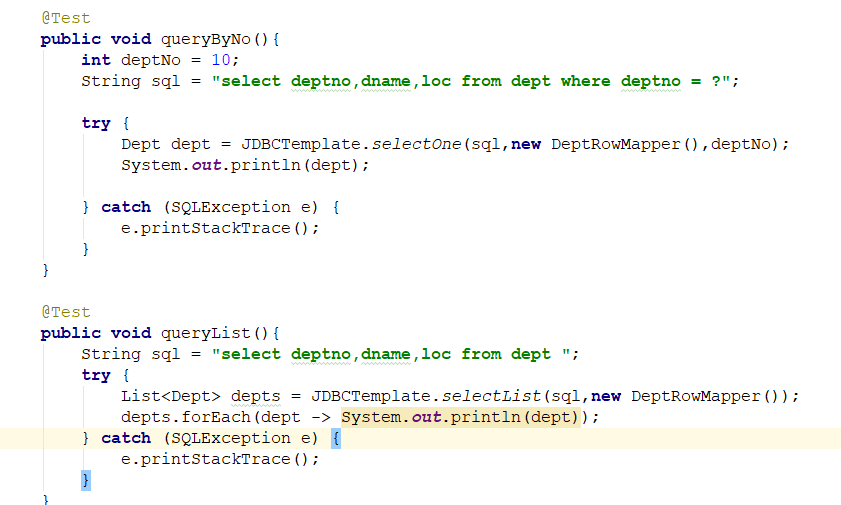
优化---- 拆分 查询集合，查询单个(内部做好判断)



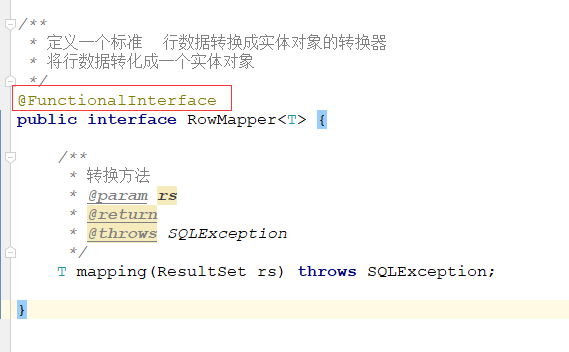


取值前，先判断，有值才能取

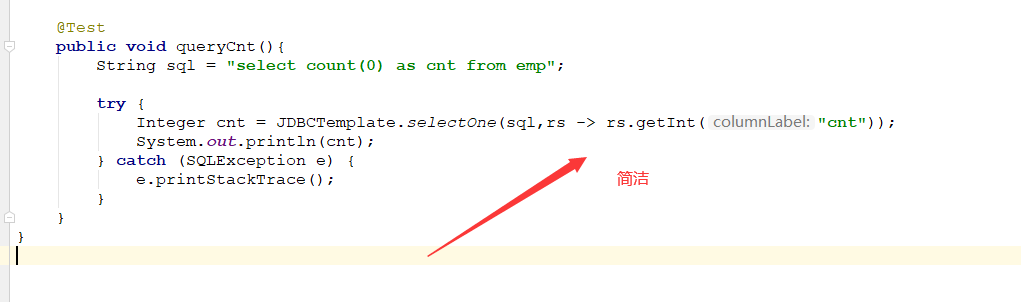




优化：转化器接口设置成函数式接口，可以使用Lambda表达式



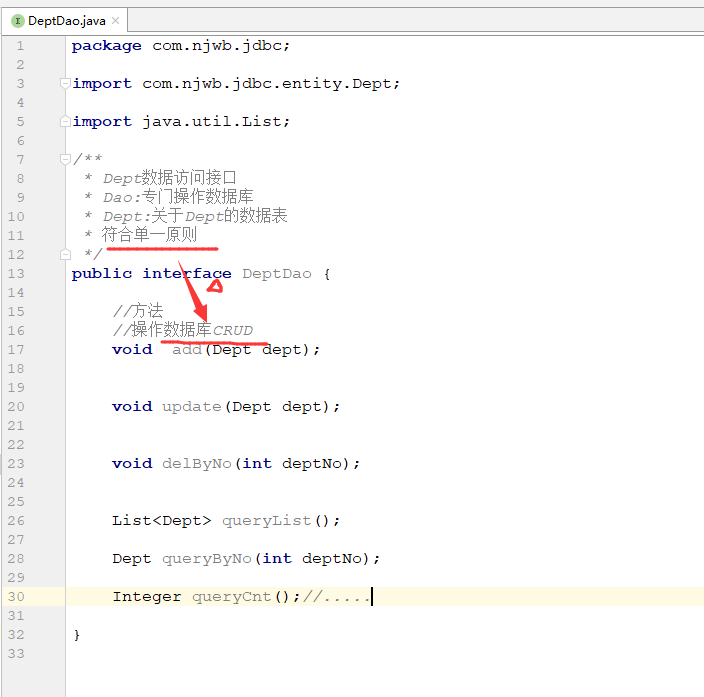
需求：统计员工总数



# 5.经典的DAO模式

DAO:data access objects 数据访问对象，专门负责操作数据库。

是面向接口编程



数据库具体是谁？确定吗？

不确定，即使使用Oracle数据库，未来可能会变吗？可能会变成mysql?

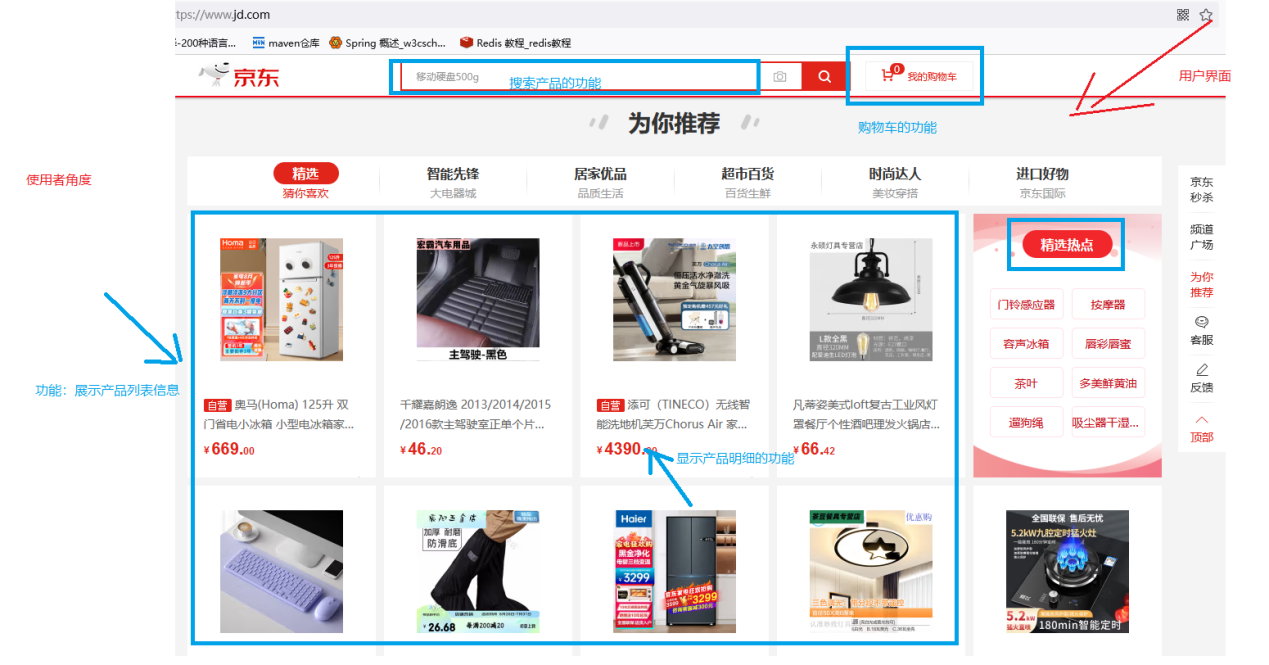
Oracle和MySql有差异 ---->代码实现不一定一模一样

如果不用接口，程序方便替换？

DAO模式是程序连接数据库的桥梁，传输的数据都是以实体为载体。

问题：DAO设计好了以后给谁用？

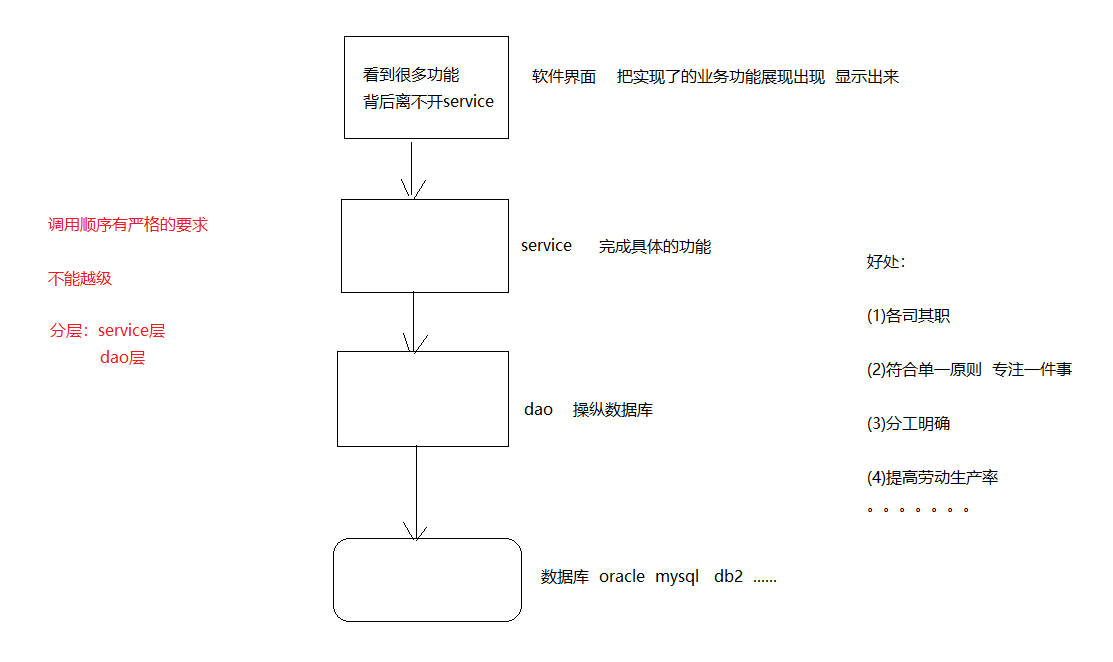
给调用者用。



面向用户的：软件的功能

软件的功能就是业务，业务和功能相对应----Service (服务、业务)

业务的具体实现需要数据做支撑--->业务调用数据--->Service调用Dao--->Dao面向接口--->Service面向接口



引入了DAO模式，从代码层面如何体现？

举例：开发一个OA办公自动化系统，至少有部门管理功能

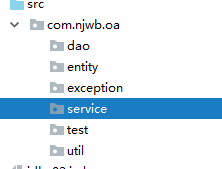
部门管理功能：部门的添加、删除、修改…….

搭建程序骨架，实现

(1)创建项目

(2)导入相关的jar

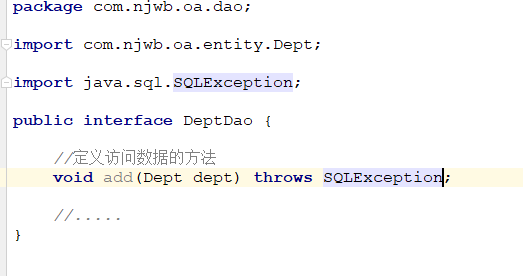
(3)创建合理的包名 ----体现出分层的概念



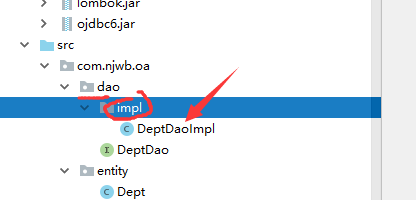
(4)封装好的工具放到util中

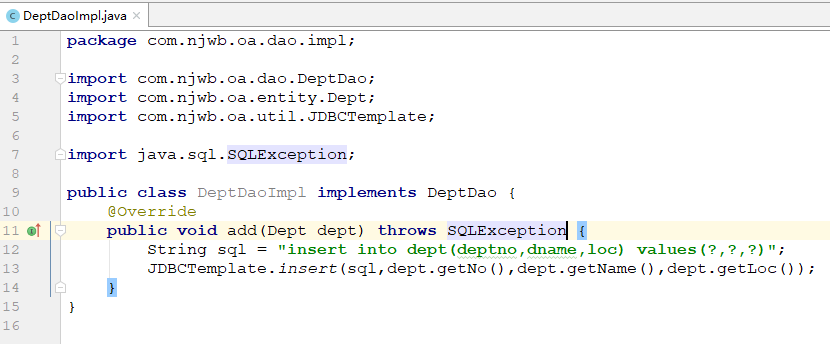
(5)定义实体类Dept

(6)定义DAO接口 DeptDao



(7)定义DeptDao实现类 DeptDaoImpl

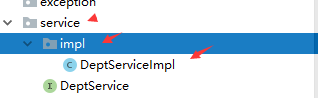


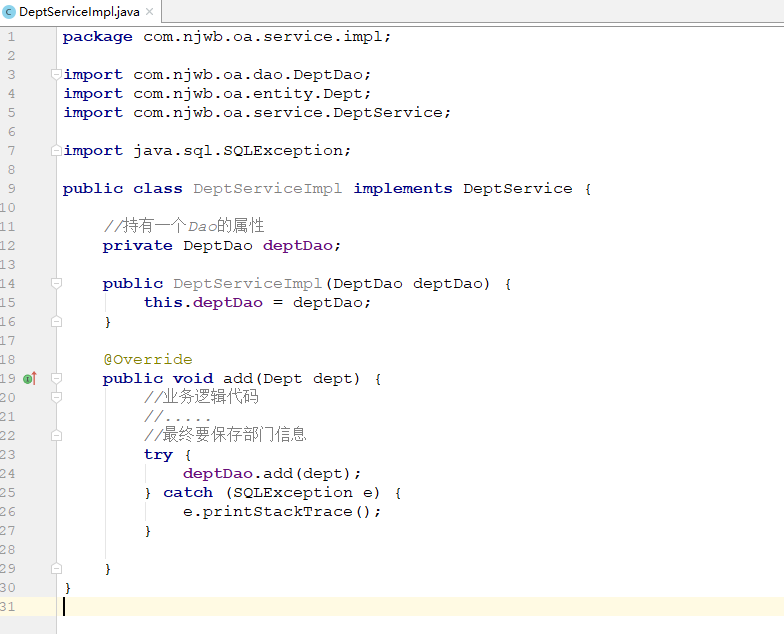


(8)定义DeptService接口

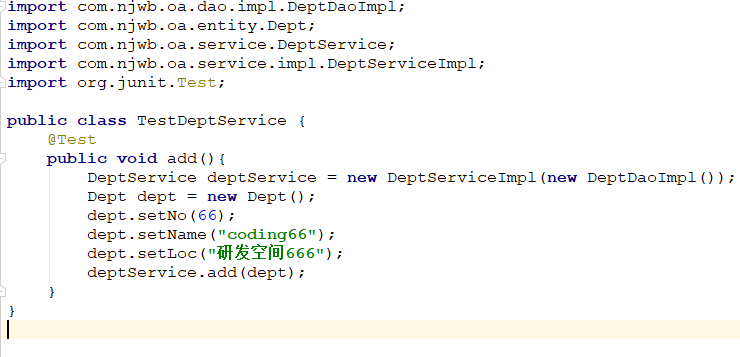


(9)定义DeptServiceImpl实现类





(10)测试业务功能



# 6.加入事务

需求：部门添加的时候，记录一条日志。

业务 ----->Service

一个业务 ---->一个Service中的一个方法

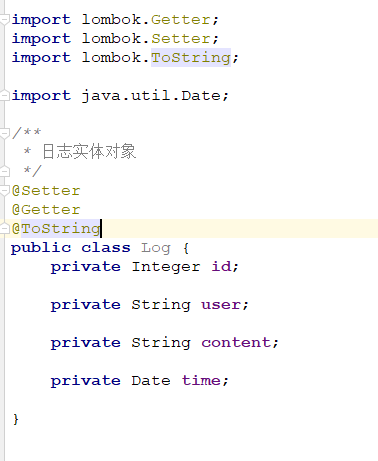
部门添加 ---->DeptService.add{

保存部门信息 ---->DeptDao

保存日志信息 ---->LogDao

}

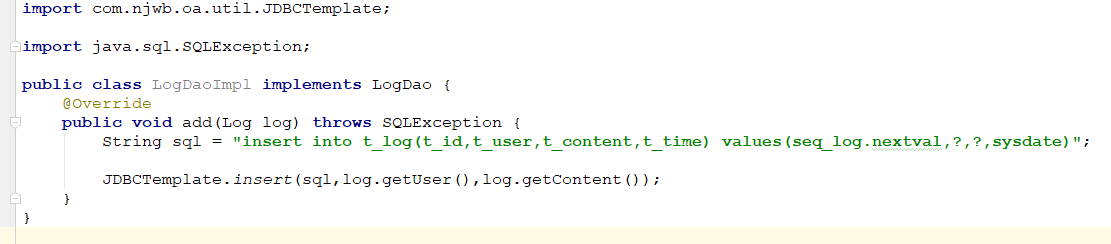
(1)实体类Log



(2)LogDao接口



(3)LogDaoImpl实现类



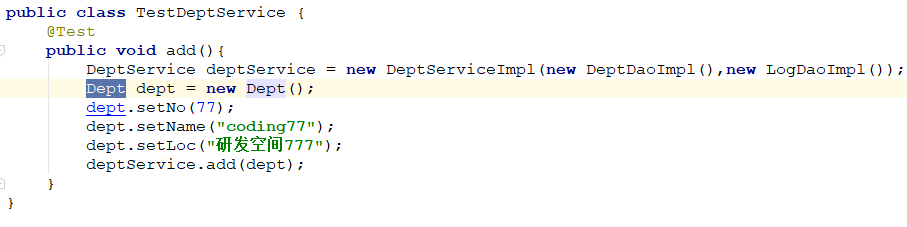
(4)测试LogDao

省略….

(5)LogService中调用LogDao



(6)测试DeptService

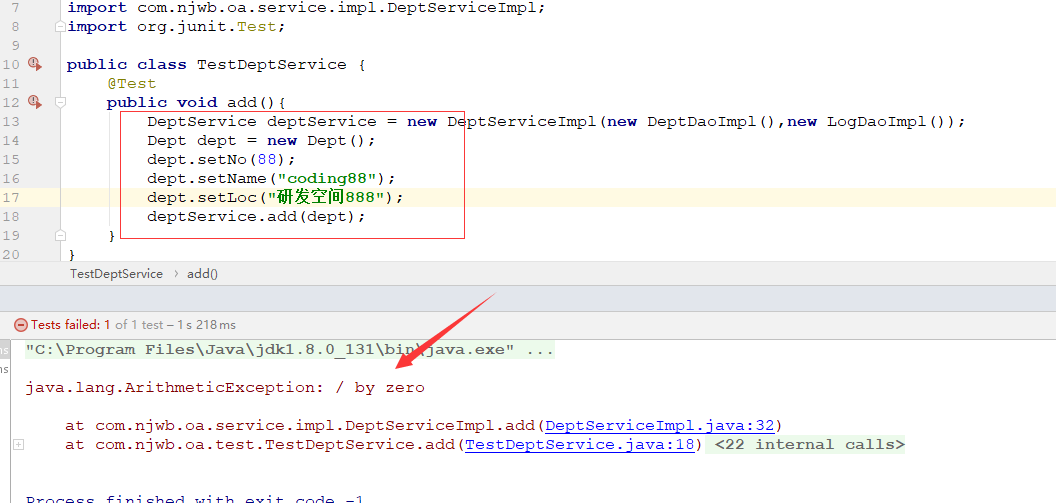


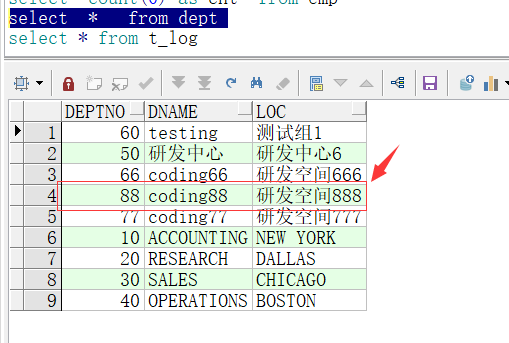
小结：

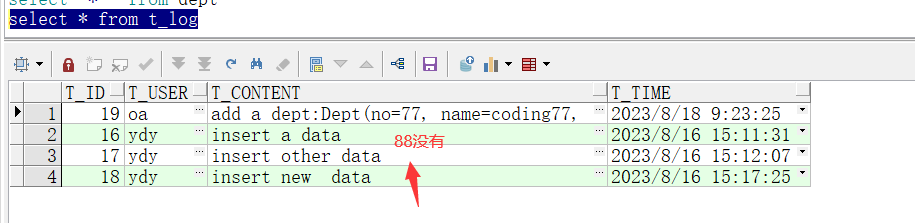
一个service方法，可以调用多个dao方法来做数据支持。

观察现象：









从代码的运行过程分析：结果数据没有问题。

但是：

如果把add方法想象成转账功能

deptDao.add(dept) 想象成 一1000万

logDao.add(log) 想象成 + 1000万



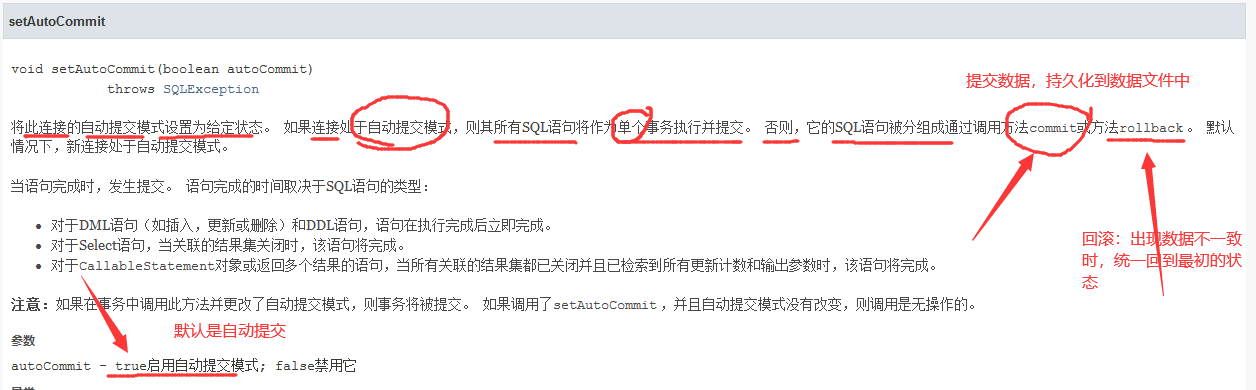
从逻辑的角度：造成的数据不一致问题。----事务问题

何为事务：

将多个操作当成一个整体，是原子性的操作，是不可分割的，要么全部成功，要么全部失败。

目前代码不支持事务问题：

不支持的原因：





解决办法：

只有一个

将自动提交设置为手动提交

a.connection.setAutoCommit(false)

b.connection.commit() 程序正常执行完毕时，提交事务 数据做永久性的修改

c.connection.rollback()程序执行异常时，回滚事务 数据撤销修改，回到最初的状态。

思路：



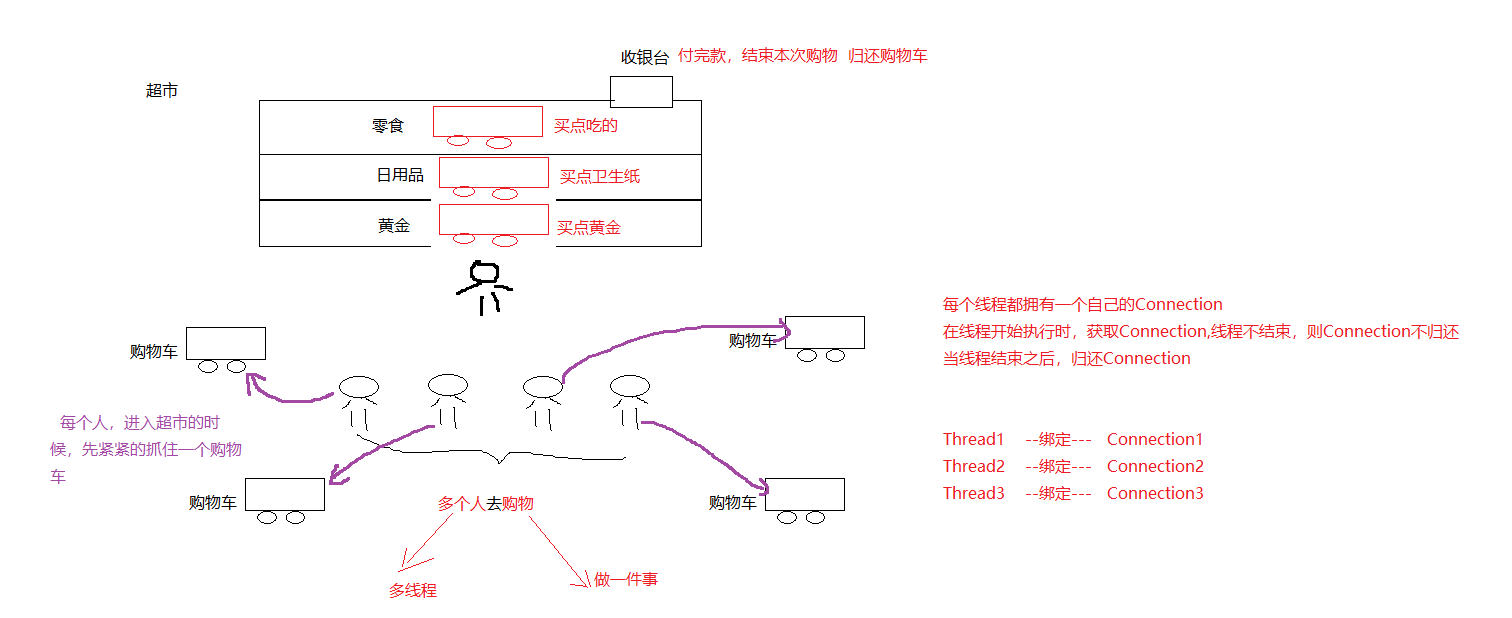


应用事务



前面的做法，全部错误。

当务之急：保证多个地方的Connection是同一个，相当的困难。



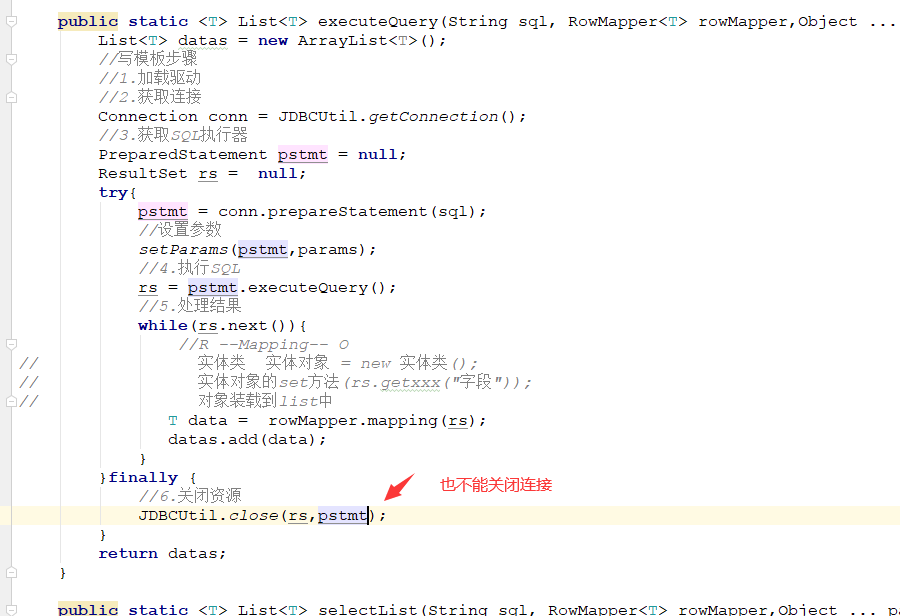
如何让线程和Connection对象相绑定？

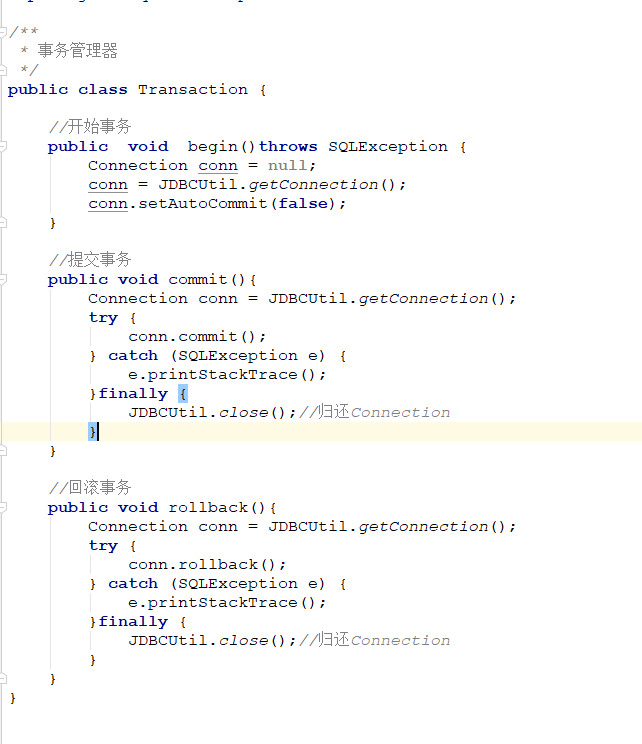




JDBCTemplate







应用

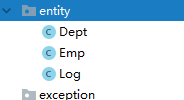


测试：当出现异常时，一定要回滚数据，保证数据的一致性。

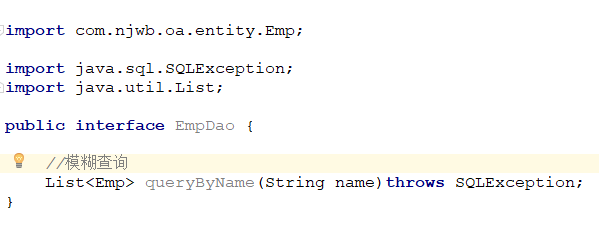
# 7.模糊查询

根据员工姓名模糊查询

(1)定义Emp

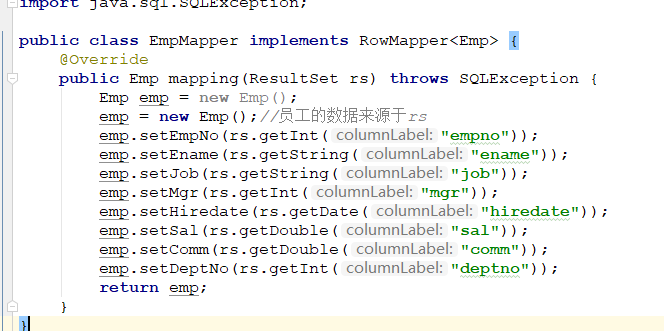


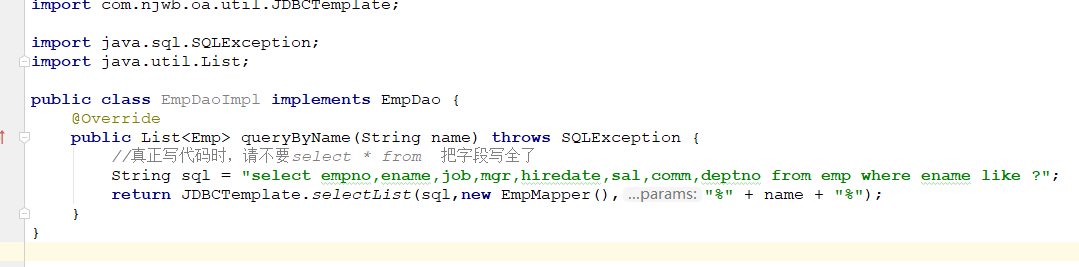
(2)定义EmpDao



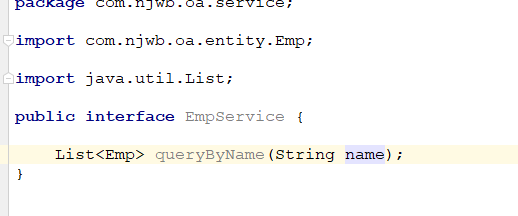
(3)定义EmpDaoImpl



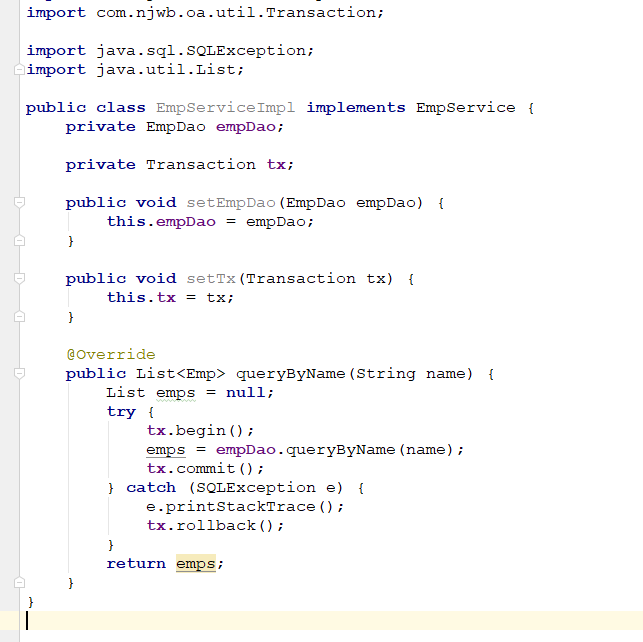




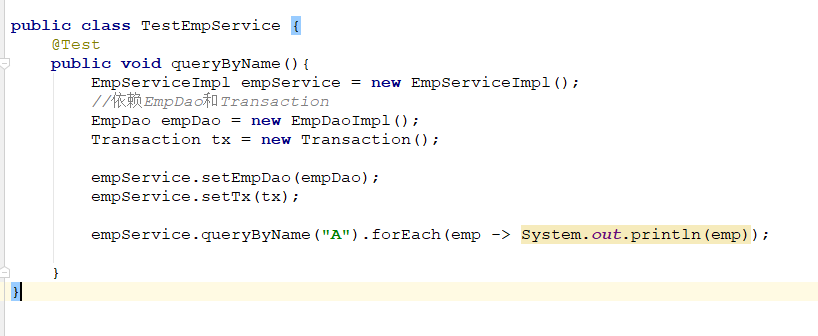
(4)定义EmpService



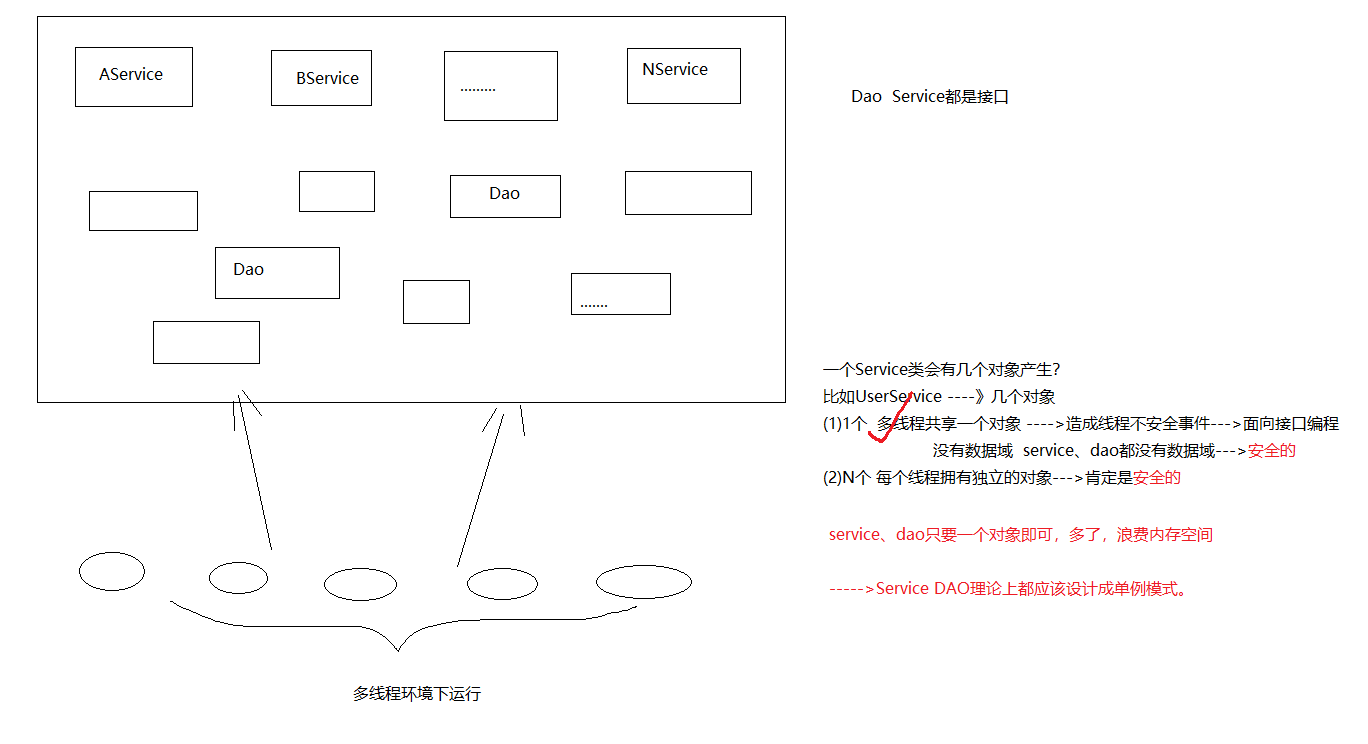
(5)定义EmpServiceImpl

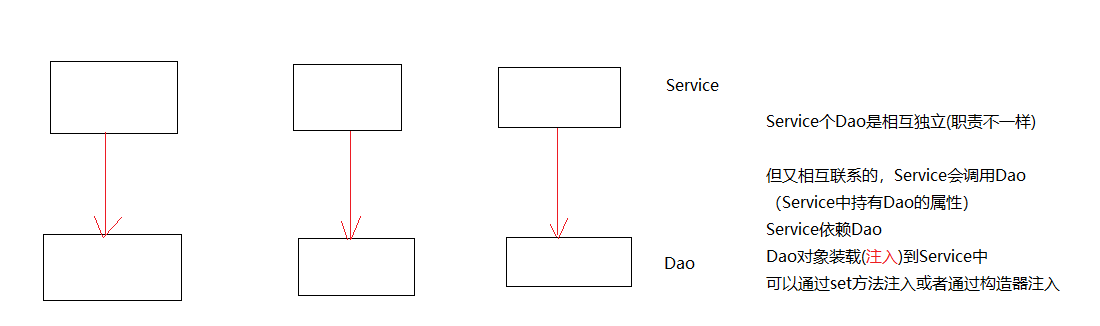


(6)测试



# 8.对象工厂





程序是有多个Service和多个Dao协同完成，所以：Service、Dao都是程序的基础组件

程序在正式运行之前应该将所有的组件创建好，并且按照一定的规律组装好。

为了实现上述的方案

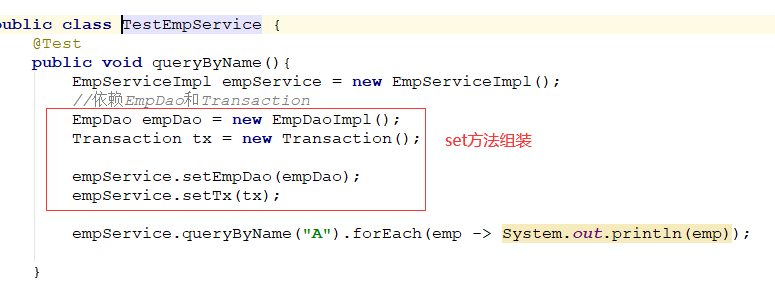
(1)将所有的service和dao全部设计成单例

N个Service, M个Dao都干成单例 ---->麻烦

(2)依次组装

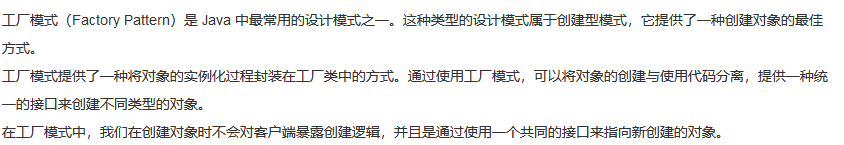
手动组装---->麻烦

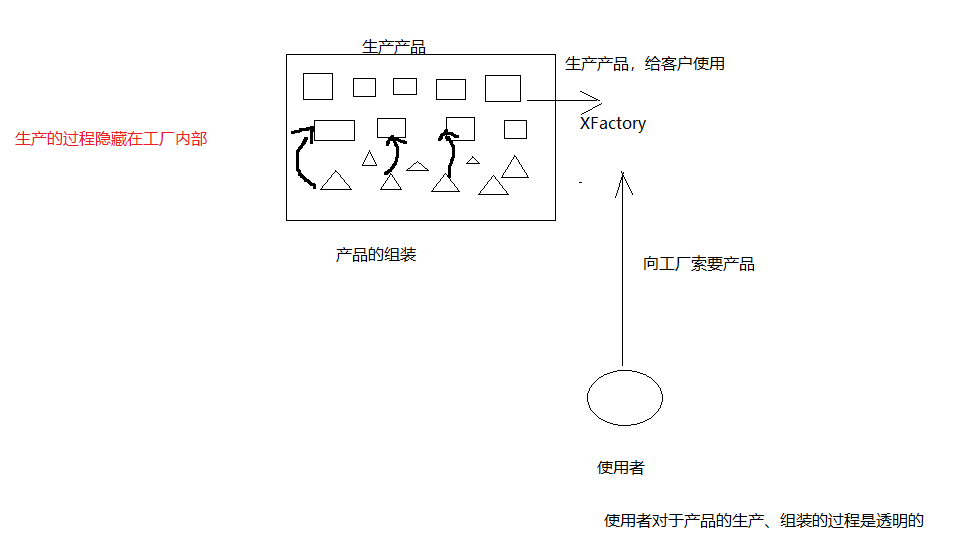




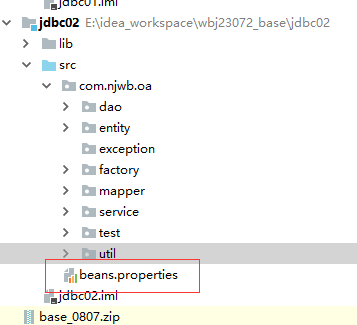
迫切的需要自动化程序：帮助一次性创建好所有的对象(1次)，对象之间的组装 ---->工厂模式

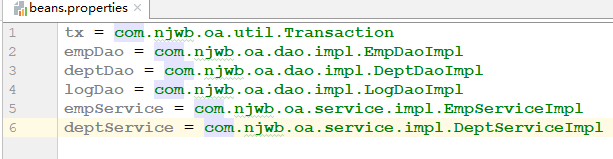
什么是工厂模式：





设计配置文件，统一将配置文件定义在src/目录下









测试



使用该工厂有几个注意点：

(1)所有的Bean都必须拥有无参构造

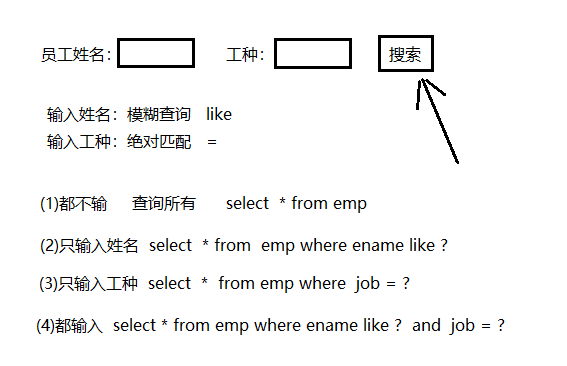
(2)属性都是通过set方法设值

(3)保证属性名称和配置文件中的id名称保持一致

# 9.综合案例

## 9.1 多条件查询

需求：根据员工姓名、工种多条件查询员工信息



实现的方案

(1)穷举法

列举出所有的可能性，通过判断 --->构造不同的SQL语句 --->分别实现。

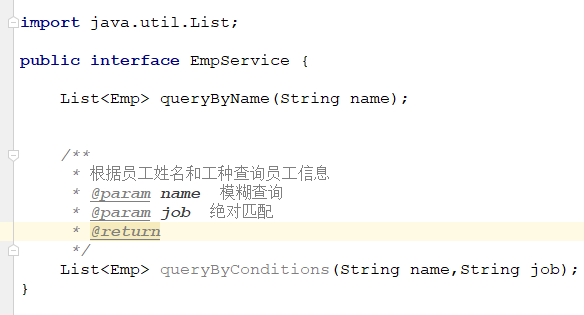
如果可能性比较少，可以采用，但是如果可能性太多，此种办法不合适。

(2)巧用恒成立 1 = 1

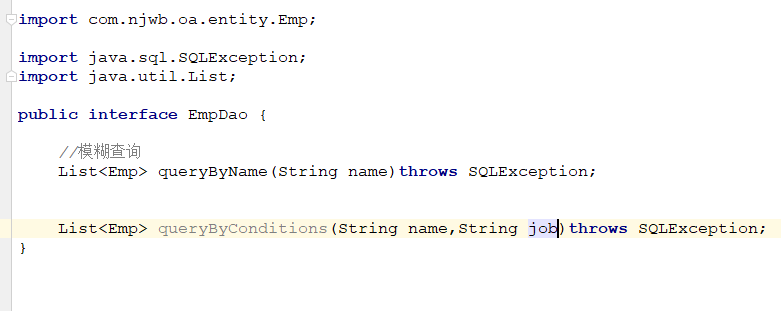


只要判断输入的条件是否为空，如果不为空，追加一个条件 and

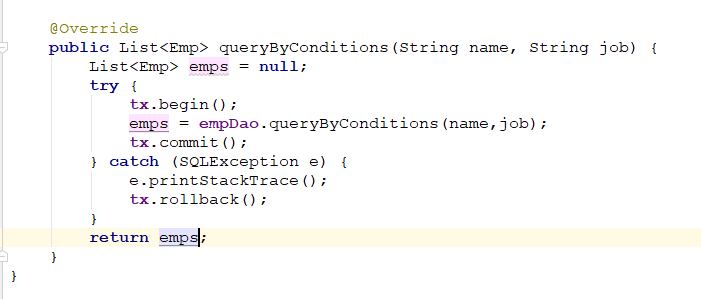
定义EmpService接口方法



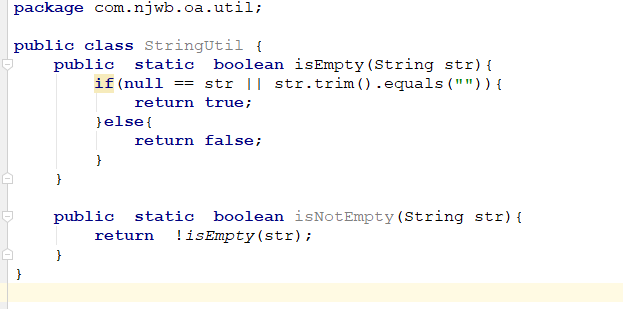
定义EmpDao接口方法

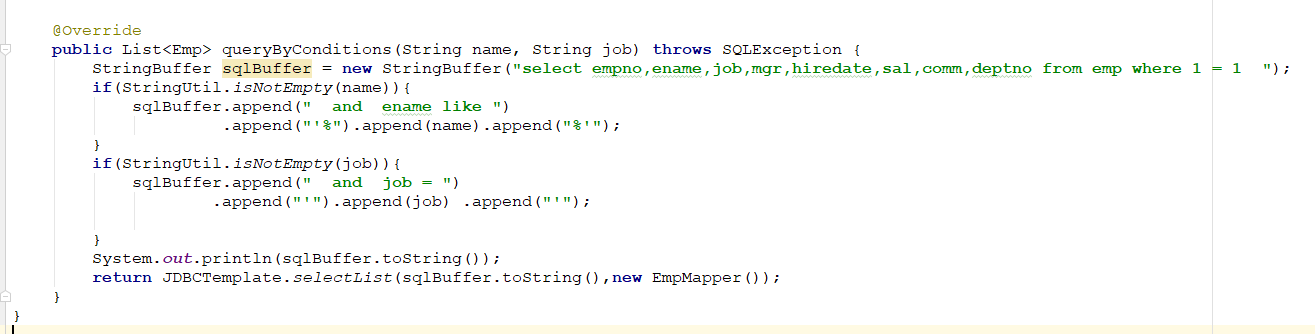


实现EmpService接口方法

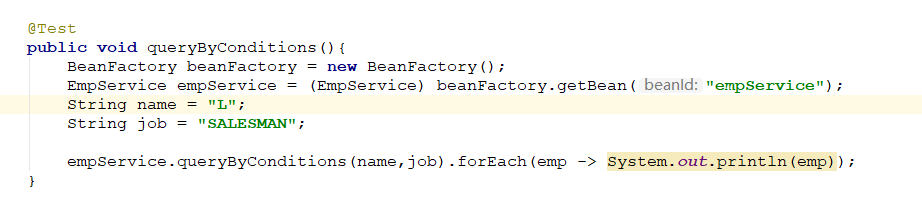


实现EmpDao接口方法





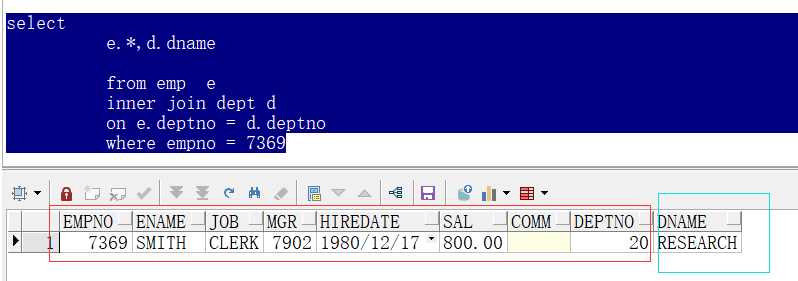
测试



## 9.2 多表查询

需求：给定员工编号，查询员工信息及员工所在的部门名称

SQL：



查询的数据来源于两张表，Java中采用什么对象来接收？

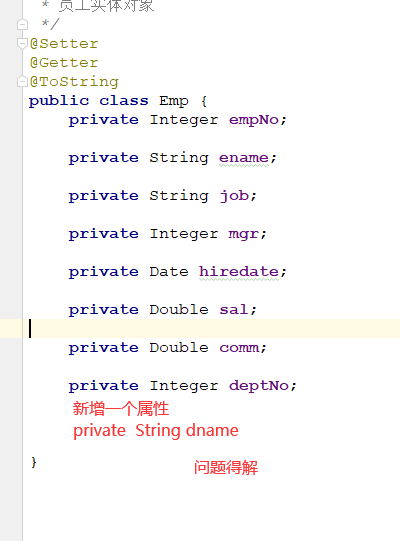
表与表之间有关系，通过外键来描述--->关联查询

本例：主要查询Emp信息，带出部分的关联数据-----》Emp对象接收

问题：Emp中没有部门相关的属性数据，因此不能完整的接收上述的数据。

怎么办？

办法1：缺少dname字段对应的属性，在Emp中增加一个属性



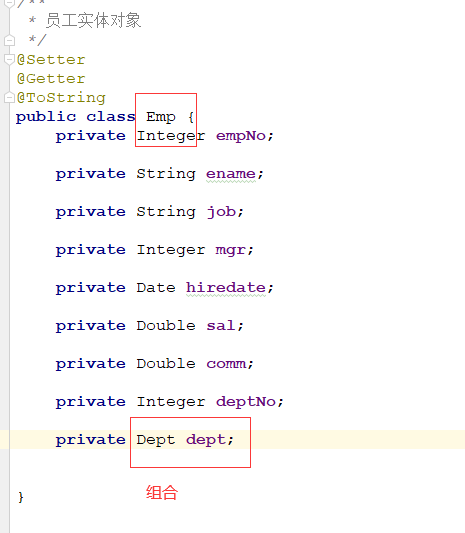
问题：与实体类的概念相违背。emp表中有一个字段叫dname?

dname本应该属于Dept的实体，就应该存在于Dept中。

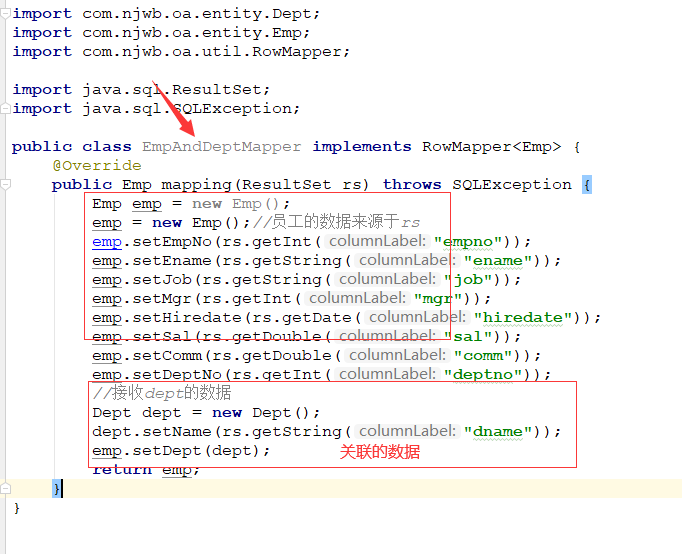
方法2：表与表之间有关系 ---->实体与实体之间的关系

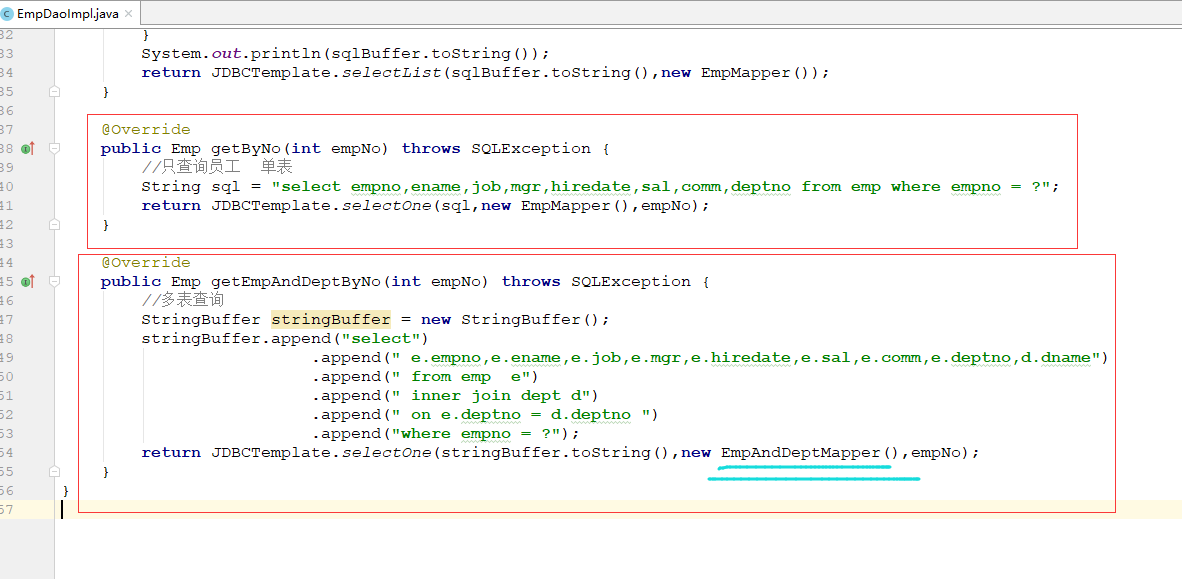
表与表之间的关系采用外键表示

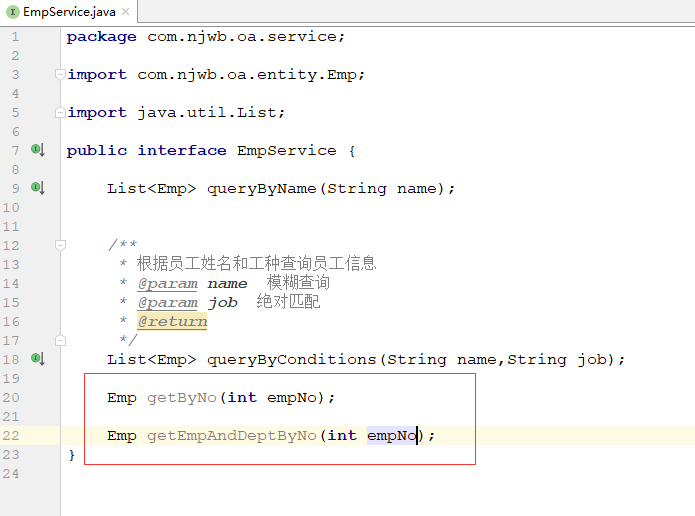
实体与实体之间的关系，采用的组合表示

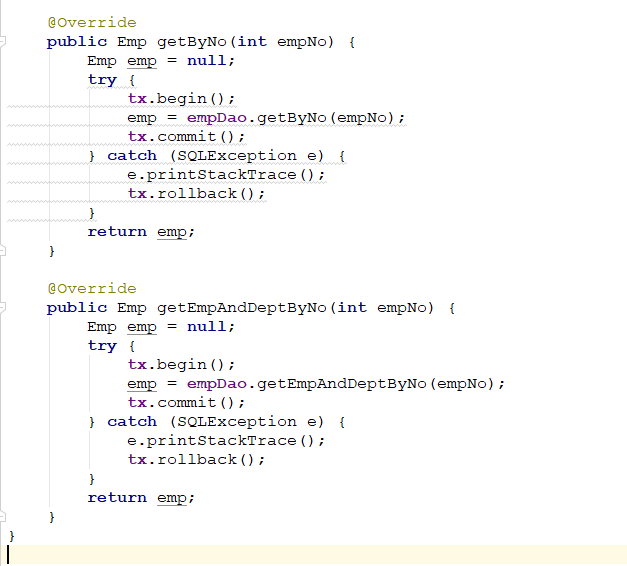




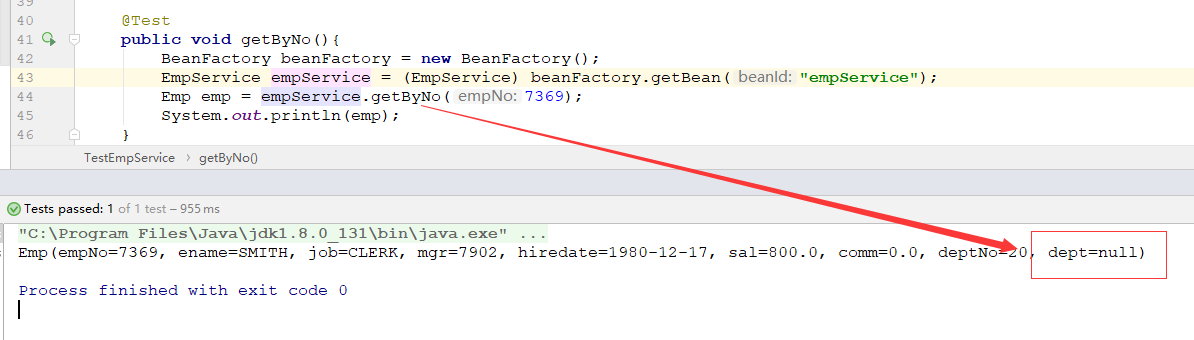








测试





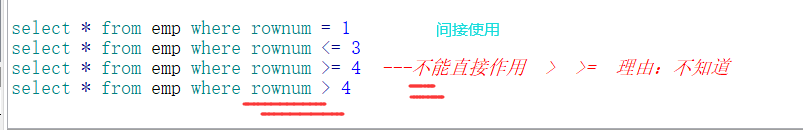
## 9.3 分页查询

需求：按照员工的工资降序排列，分页查询数据

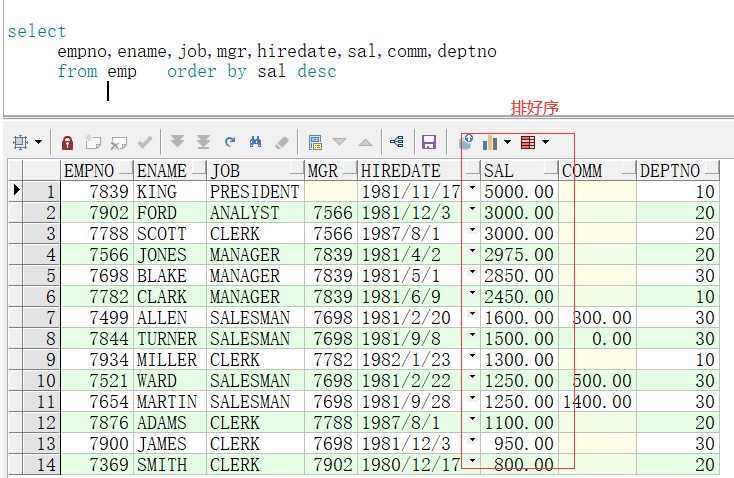
每页显示5条件  
SQL语句：

借助rownum关键字

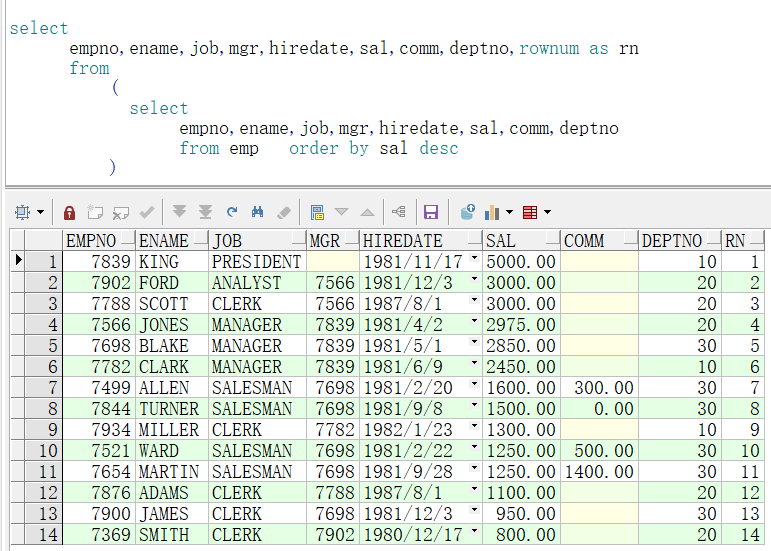
但是：rownum不能直接参与排序 ----->子查询

 也是使用子查询。

(1)处理原始数据，包括过滤、分组、排序

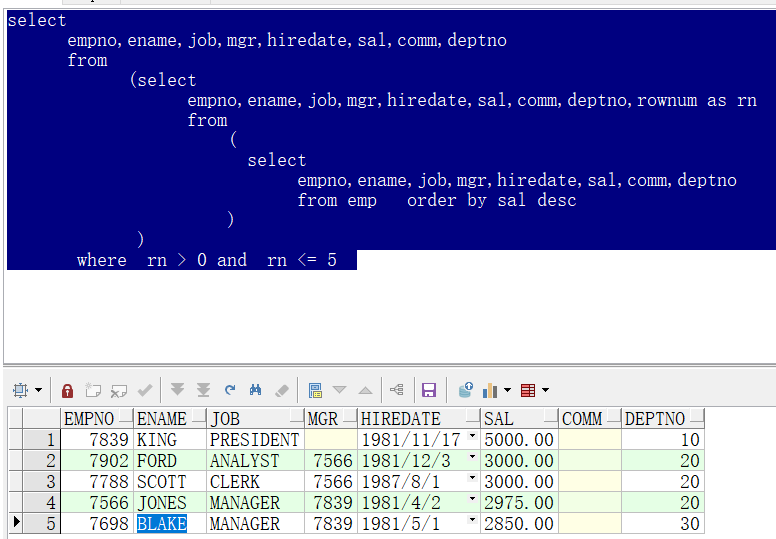


(2)只带出行号，避免直接使用rownum做 > 或 >= 的计算

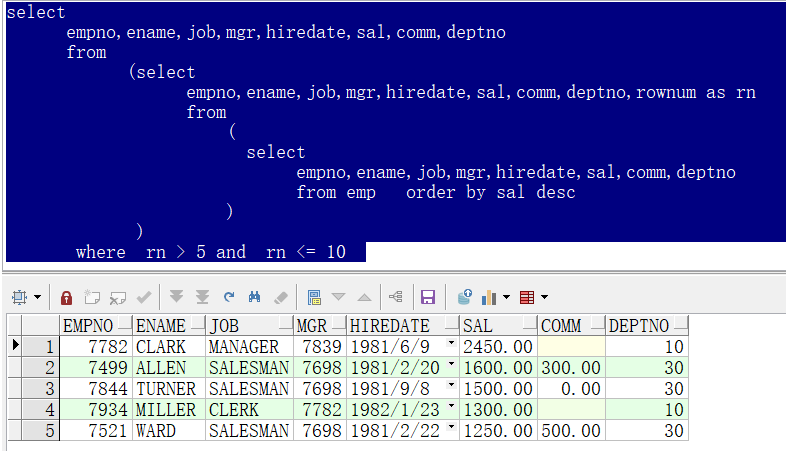


(3)利用rn 选取区间元素

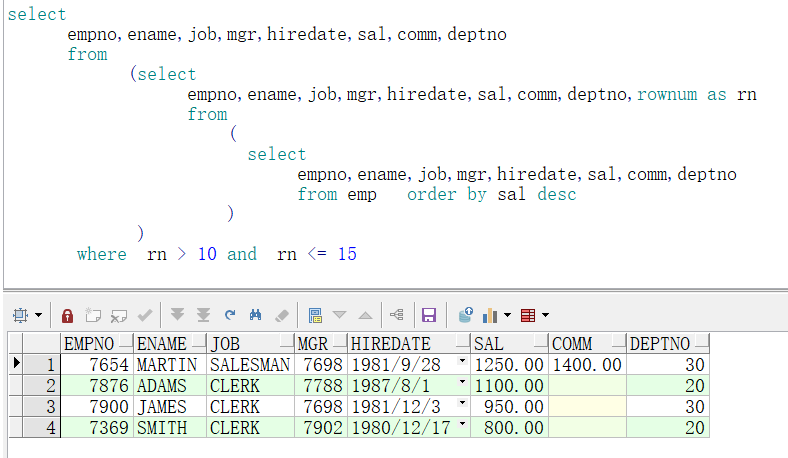
第一页数据



第二页数据



第三页



分页规律：

rn > min and rn <= max

设每页显示的数量为pageSize = 5

当前的页码为pageNo

pageNo = 1 ---> min = 0 max = 5

pageNo = 2 ---> min = 5 max = 10

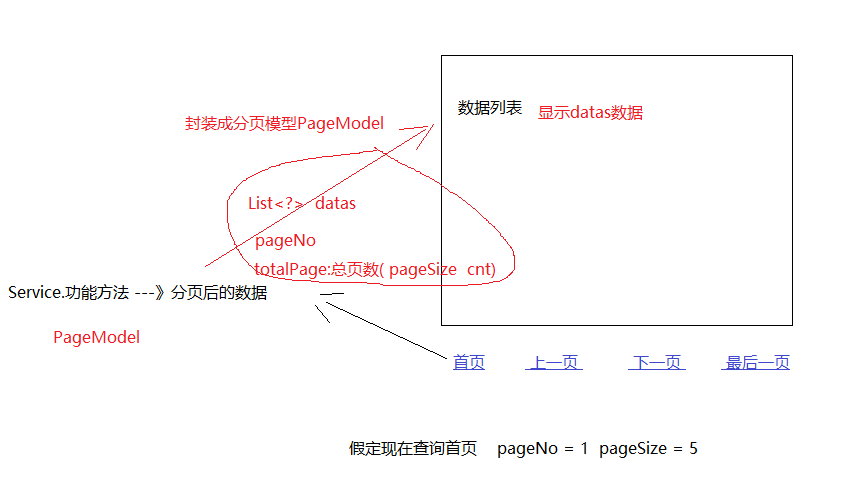
pageNo = 3 ---> min = 10 max = 15

min = (pageNo - 1)\*pageSize max = pageNo\*pageSize

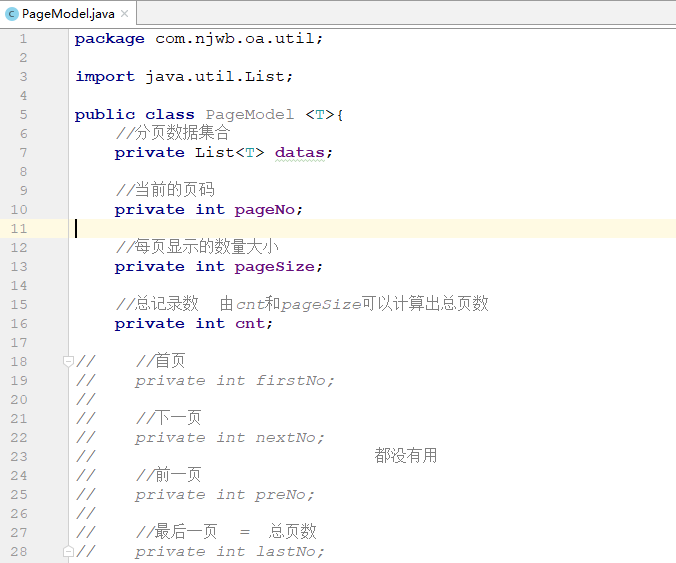
rn > (pageNo - 1)\*pageSize and rn <= pageNo\*pageSize

结论：分页和当前的页码以及每页显示的数量有关。

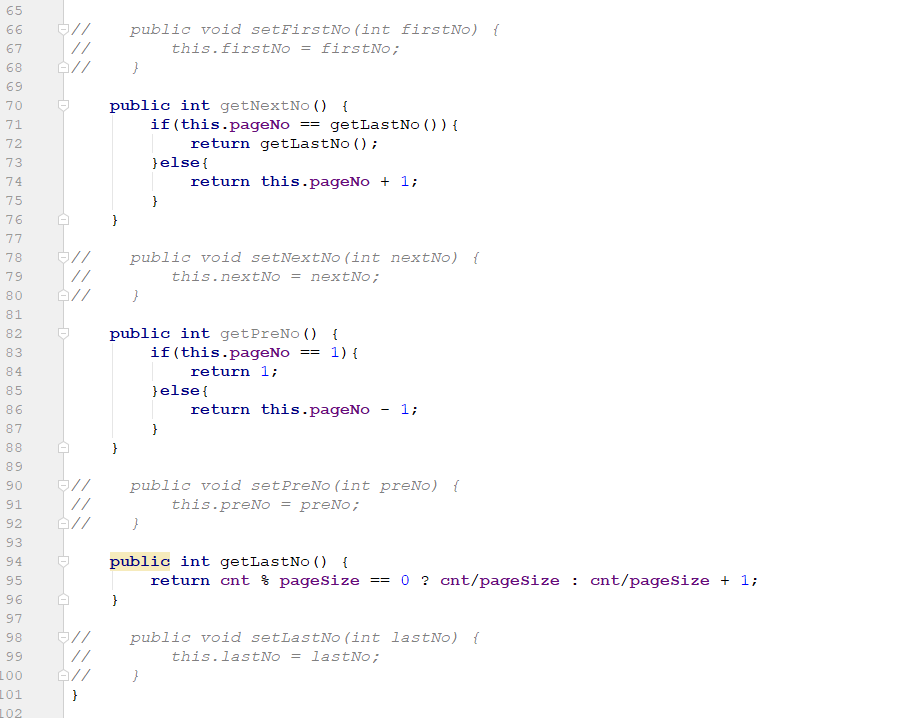
分析：业务方法如何设计？



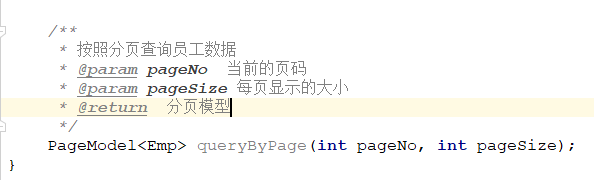
封装分页模型



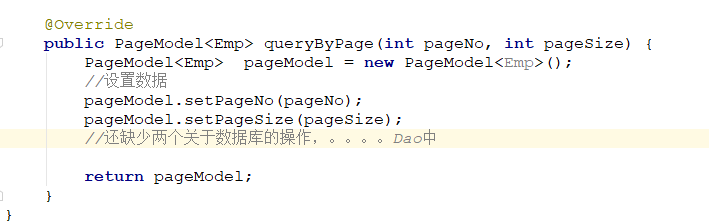




在EmpService接口中定义方法



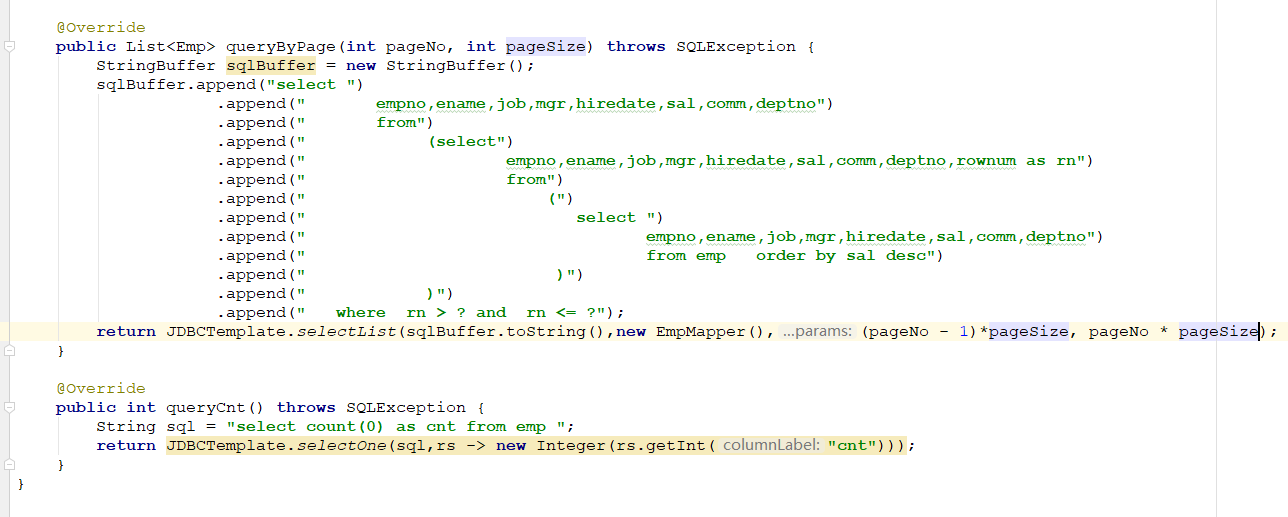
EmpServiceImpl实现方法



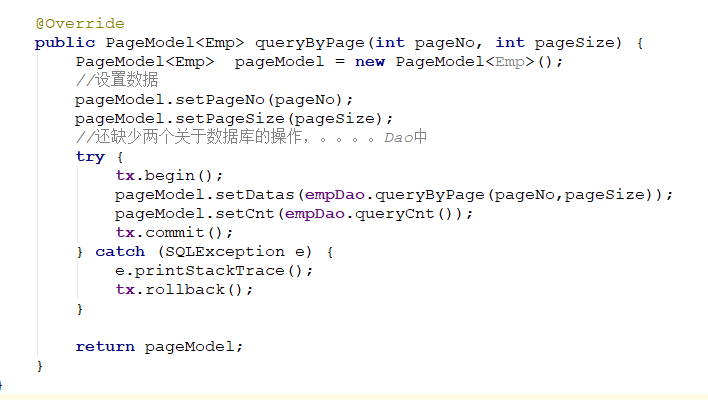
EmpDao



EmpDaoImpl



完善service



测试

