# 1、数据架构

## 1.1 数据库ER模型

PowerDesigner ER建模

数据库的ER建模，ER，Entity Relationship，实体关系，一个表就是一个实体，设计表与表之间的一个关系，也就是实体与实体之间的一个关系

包括每个实体包含哪些属性，表包含了哪些字段

一般就是根据需求分析以及概要设计中的内容来考量，就是说负责的这个模块，具体需要哪些数据库表去承载，才能实现出来

将所有需要的表，全部在一张ER模型图中体现出来

## 1.2 数据库逻辑模型

PowerDesigner将ER模型转为逻辑模型

## 1.2 数据库物理模型

PowerDesigner将逻辑模型转为物理模型

在正规的流程中，必须是对每个系统，都完成数据库ER模型之后，去完成后续的设计，包括接口、类图、运行流程，结合运行流程，你要执行哪些增删改查的操作，反过来完善物理模型的设计，包括索引、主外键关系

但是我们这边说一个实践中可以权衡的这么一个方案，就是说，实现类图这一块，我们将增删改查的controller、service、dao、mappper，都没在类图中去设计。这个在实际开发中是可以的。

所以这就导致我们没法在现在这个阶段，就直接完成索引的设计，因为包括mapper里面有哪些方法和sql都不太清楚，就可以在开发的过程中，一边开发，一边结合你写的sql来设计数据库的索引。

然后在code review的时候，申请别人review一下索引的设计。

# 2、接口

（1）DO：Data Object，数据对象，一个DO一般是对应于一张表，或者是Mapper查询接口中的返回类型，mapper和dao返回的都是DO

（2）DTO：Data Transfer Object，数据传输对象，service组件需要将dao组件返回的DO对象给封装成DTO对象，再返回给controller组件，或者是别的service组件。service必须将数据封装为DTO，返回给其他的controller或者是service来使用

（3）VO：view object，其实就是将页面中需要展示的数据封装在VO里面，然后在以前是将VO给传输给页面模板，比如JSP（JSTL标签语法，可以将VO中的数据给展示出来），freemarker，velocity，thymeleaf。restful接口，在接口中可以将一个接口要返回的数据，封装在VO里面，从接口返回即可。

（4）Query：查询条件，controller里面接收的页面发送过来的查条件，哪怕是service层对外暴露的接口，也可以用query来接收复杂的一些查询条件

（5）PO：就是将DO不是作为data object，而是作为domain object，然后将PO作为persistence object，一个PO对应一张表，DO就是我们平时放在domain包中的领域模型对象

（6）BO：service组件对外可以返回这个BO，这个BO的话呢是包含业务逻辑的，不光是包含一堆数据和getter和setter方法，还包括了这个很多的业务逻辑方法

（7）AO：application object，就是跟展示层相关的一些数据

大家有没有发现迄今为止，我们耗费了大量的精力在设计这个接口文档。但是其实接口怎么设计，设计的怎么样，不是什么技术问题。接口只不过是两套系统之间互相交互的一个协议。

所以，我是这么想的，如果咱们继续耗费大量的精力在接口设计这块，可能会导致我们的课程进度较为缓慢。学习的价值不是特别大。

就是说，如果是那种面向前端的接口，controller里面的接口，我们就不在这里设计了。写很多json格式的输入和输出，很麻烦，耗费的时间太大了。咱们这块直接在详细设计这个环节给省略掉。当然了，如果你们做项目，一定要按照我之前几讲的示范，所有的接口都在详细设计文档里来写。

但是我们这里，就是说对前端的接口，我们在这里不写了，后面写代码的时候，咱们直接就把接口定义出来，然后就直接写代码。

但是，有一点很重要的是，这个模块与模块之间的交互的接口，比如说采购中心和调度中心之间的交互，都是需要依靠接口的，而且我们需要在整个工程初始化的时候，就将这个接口给定义出来。所以模块与模块之间的交互的接口，必须是要设计的。

## 2.1 XX模块

### 2.1.1 XX接口

（1）接口类

（2）接口方法名称

（3）接口描述

（4）输入

（5）输出

# 3、开发架构

## 3.1 实现类图

根据之前的需求分析文档、概要设计文档、数据库模型、接口，去考虑，实现接口的功能，需要哪些类，类与类之间的关系是什么，每个类有哪些成员变量，每个类有哪些方法

在这里说明一点，这里其实就是所谓的Java面向对象编程的核心所在，这里其实是很体现一个工程师的水平的

UML的类图里，其实就三点

（1）把每个类都画出来

（2）类与类之间的继承关系画出来，实线+空心箭头

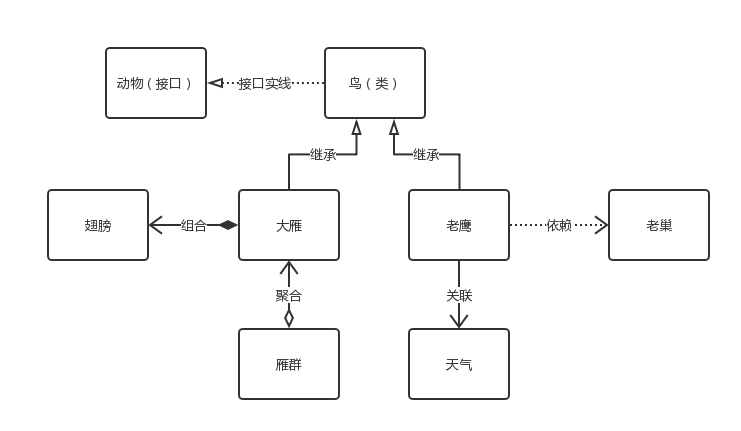
（3）接口实线关系，虚线+空心箭头

（4）依赖关系，虚线+简单箭头

（5）关联关系，实线+简单箭头

（6）组合关系，实心菱形+实线+简单箭头

（7）聚合关系，空心菱形+实线+简单箭头



根据我的一个实践，我有这么一个经验，就是说如果要提高开发效率，在这个实现类图这个环节，可以做一定的简化。

在一个普通的业务系统里面，一定是有一些最最基本和最最简单的CRUD，增删改查，如果是最普通的增删改查的操作，可以不用画这个实现类图。此时，就是对于这种CRUD的功能，只要有这个接口，运行流程，你知道基于设计好的数据库模型，可以实现就ok了。

## 3.2 包设计

在UML中有包图的概念，我在这里先补讲，我比较少用

至少你要设计出来，你有哪些包，每个包下面有哪些类

com.zhss.eshop.order.domain

com.zhss.eshop.order.controller

com.zhss.eshop.order.mapper

com.zhss.eshop.order.dao

com.zhss.eshop.order.service

# 4、运行流程

如果是有较为复杂的业务流程，则必须画活动图

如果仅仅只是一些相对简单的CRUD，则文字描述清楚你的代码实现逻辑即可

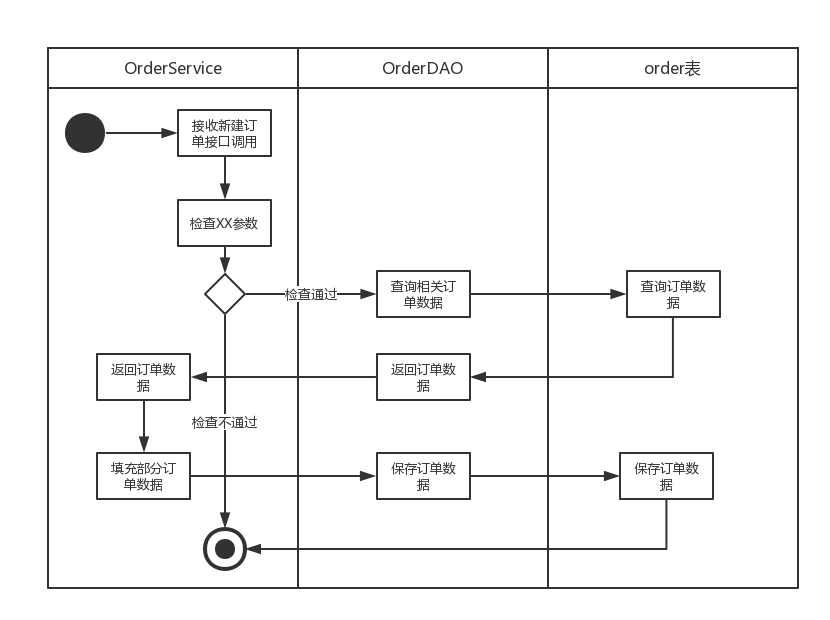
## 4.1 XX模块

### 4.1.1 XX功能1

每一个功能，参考之前概要设计中的每个功能的用例图，类图、数据库表，画出来这个活动图

用例图，基本上还是重点在体现较为高层的系统功能的运行流程

活动图，就要画出来，每个类和每个表之间的交互关系，完整体现出来你的功能要实现的时候，各个类以及表之间的交互顺序和逻辑



### 4.1.2 XX功能2

# 5、测试用例设计

## 5.1 单元测试用例设计

### 5.1.1 XX模块

5.1.1.1 测试用例1

对每个类设计对应的单元测试类，每个方法都要有单元测试的方法

（1）数据准备

（2）输入

（3）输出

（3）资源清理

5.1.1.2 测试用例2

## 5.2 冒烟测试用例设计

### 5.2.1 XX模块

5.2.1.1 测试用例1

（1）数据准备

（2）输入

（3）输出

（3）资源清理

5.2.1.2 测试用例2

# 6、日志设计

## 6.1 XX模块

### 6.1.1 XX功能

（1）日志埋点1

（2）日志埋点2