# 模拟面试卷四

## ERP进销存项目

### 1.说说你最近做的这个项目的背景？

这个项目是为××（名称自己构思，为防止同学们面试期间撞车，此处不提供公司名称，但是此名称需要同学们调查一下这个公司是否真实存在，在百度上搜索一下，例如武汉商贸公司，列出来一些名字，不要找第一页的名字，最好向后翻几页）商贸公司做的，公司原有的项目是一个覆盖地区为省的平台，现在公司规模扩张，全国各地建分公司，省内管理相对来说比较有局限性，加上全国全联网，采购、销售、货运、仓储、配货、财务核算各方面发生了很大的变化，所以重新起了一套新的平台。

这个项目可以说算是二次开发，但是业务里程变化蛮大的。尤其是加上了一些数据分析，数据挖掘的东西，而且报表的统计数据量也比之前的大很多，复杂度高很多。这个项目开发主要是从CRM、进销存、仓储、财务四个方面做了大的改动，基础平台方面多添加了一些功能，比如像合同啊，报销啊这些东西不再以网点为单位了，改成全国统一编排。

### 2.你的表结构是如何设计的？为什么这么设计？

表结构方面初期受原始项目的影响，一直想着在原始项目基础上进行修改，后来发现原始项目的库设计有问题，进行改造相对来说工程量很大，而且关联关系新设计中发生了比较大的变化，最后推翻了重做的，原始数据做了个数据迁移。

比如说省内业务时，供应商A处采购，到货运中心后，然后发货给B客户，但是改造成全国业务后，供应商A处采购如果再到货运中心可能会造成成本的提升，需要直接发往B客户处，这个过程中运单就在走单过程中原始项目无法处理了。这里供应商A的单号是采购单号，但是到客户B那边应该是供货单号，需要做单号的更改处理。最后设计的将原来的分销业务扩展成了分销和直销两种。表结构方面也做了调整，将原始的业务扩展成分销表与直销表，分开管理。同时也会后期数据挖掘降低一些工作难度。

设计方面比较独特的几个地方我说一下

第一个是供应商采购价格这个字段，原始设计中是一个商品对一个价钱，基于原材料涨价，人工涨价导致的价格变化直接对商品采购价格进行修改。做成全国项目后，采购来源就更多了，为了辅助后期的数据挖掘，比对价格涨幅与预计增长量，配合货运成本计算，需要对所有的价格变化波动进行数据分析，所以添加了价格波对表，同时原始表中也存储目前的最终价格。虽然数据微量冗余，最后价格两个表中保存，但是整体运行性能方面得到了很大的提升，而且这样设计才能配合多组数据，计算采购差价，核算最终成本。简单来说，A地的货物便宜，但是到达客户的运输成本高，B地的货物贵，到达客户的运输成本低，总体核算需要配合很多数据。但是又不能单纯的从价格方面放弃某一个供应商，所以需要监控其价格波动量，必须做相应的表设计调整。

第二个是采购运输方面的调整比较大。省内项目的中的设计允许出现一次采购途径A,B,C,D四个地方，一次完成若干个运单，也就是车务调度与采购单是一对多的关系，但是全国项目中，由于发车地点多了，若干个采购任务可以从不同地点发车，运输成本反而降低了，毕竟第一个采购完成的货物在路上跑了很久。再有，不同地点的车务组规模不同，不一定能完成一个采购任务。所以设计方面做了很大的调整。一个采购单可以允许拆分成若干的子任务单，而每个车务调度只对应一个子任务单。简单的说就是原始车务调度对采购单是一对多，改成了采购单对子任务单一对多，车务调度对子任务一对多。降低了原始任务中一个运单运输时间成本过高的问题。

第三个是采购执行单的修订。原始设计中采购计划单与采购执行单就是一个单据，审核中就执行。全国项目中，一份采购计划对应的采购如果一个供应商不能满足，需要从多个供应商处进行调配，这样采购执行单对采购计划由原始的一对一，变成了多对一。实际执行时，所有的采购单说的都是执行单。

还有很多此类的设计更改，我就不一一说明了。

### 3.这个项目为用户提供了哪些服务？包括哪些模块？

从大的方面来说包含有4个系统，**CRM系统、进销存系统、财务系统、基础平台**。公司的基本办公在基础平台中，CRM系统做供应商管理、客户管理、委托运输客户管理和政府部门相关管理，进销存系统包含采购、销售、库存、合同、任务委托、还有几个我涉及的不多，记不清了，财务系统包含内部结算、外部结算两大块。本来建议客户买一套用友的财务系统的，后来没有谈成，最后这一块外包出去了。其实系统中让我来看比较特色的是风险评估和数据挖掘后的预算决算处理。但是这一块不是我所在的组做的，了解不是特别多，就知道个大概，就是数据分析后，为客户直接出多个方案，供用户选择最终的执行方案。

### 4.你承担这个项目的哪些核心模块

我开始在CRM组，做了大概两个月左右，后来调到进销存那边去了。在进销存这边做了有将近4个月。进销存的主核心我都参与开发了。

我说说采购吧

首先采购发起可以从4个任务开始。1，需求采购，2被动采购，3，计划采购，4，预警采购。需求采购就是主动提出的采购，由采购部业务人员主动发起，没有前置条件；被动采购是销售那边如果遇到库存不足，自动转入一个采购申请的流程，进入到采购主线上；计划采购是ERP系统运行过程中，根据历年的采购数据、销售数据、库存数据动态计算出来，生成一份采购计划，这个只是个采购建议，然后转入采购主线，就是交给采购部业务人员那边进行确认；预警采购是库存量超过当时的预设临界点后发出一个采购申请计划。不管哪一种最终都汇总到需求采购那边，只不过转入方式不同。

采购申请单下好后，根据采购内容划分2级审批还是3级审批。如果是公司内部使用消耗品采购转入采购主管审批，然后行政主管审批；如果是公司业务采购，走3级审批。订单如果超过设定值，比如50万，需要4级审批。

审批完以后生成任务单号，转入后勤。后勤主管根据采购内容分派跟单人，跟单人根据采购单据和发货方式选择联系对方发货或者转入车辆调度或者委派第三方完成。如果对方发货，记录传真号，语音信息，交货时间这些信息；如果使用第三方运输，根据采购单据选择航空、铁路，转入CRM系统进行运单跟踪；如果公司内部组织运输任务，转入车务组调车和司机还有随行人员。运输任务跟单人可以进行更换。

运输任务完成后转入验收流程，验收完成后，给每个批次的货物分配一个内部管理的货物编号。

如果这个货是第三方到第三方的，也就是不经由公司仓储的话，需要对货物再进行一个出货编号的分配，采购流程结束。如果不是第三方到第三方的运输，货物编号分配完转入入库流程，入库以订单为单位。一个订单中所有货物入库完成，采购流程结束。上面说的这个事分销的采购流程，还有直销的采购流程，就是在中间有一个分任务单的地方不同，然后就是入库结算方式不同，按子任务入库的，而不是整个订单入库。

### 5.你碰到了哪些问题？你是怎么解决的？

最大的问题还是**设计问题**，实现方面都还好，设计比较头疼。如果没有前期的项目恐怕还好一些，受影响太大了。剩下的比较突出的问题就是页面的东西了，业务只要需求理解了问题都不大，页面的脚本控制应该说是消耗时间比较多的地方。整个项目做了半年多，现在的页面jquery比以前强多了。脚本方面就是多使用firebug这样的工具吧，次数多了慢慢就知道问题定位如何快速准确了。

### 6.项目最大支持多少并发量？

组内测试的时候并发大概到100左右还算稳定，到200丢包率很高，能达到40%，到300时丢包基本上在90%以上。这个项目的受众群体大概平均在线1000人左右，最大并发量大概在秒70到120左右，性能要求不是很高。

### 7.做完这个项目你有哪些收获？

应该说的比较日常的东西，就是脚本写的比以前好多了，剩下就是设计思路方面有了一个不小的提升，再有就是客户自定义字段以前没做过，这个项目的商品展示方面用了一次，感觉虽然很繁琐，但是很有意思。

* 项目课程介绍
  + **项目涉及前期课程知识点**
    - Struts2（包括拦截器，文件上传下载）
    - Hibernate3（单表数据基本操作、关联关系数据操作、延迟加载）
    - Spring3（Bean配置与管理，团队开发模式，AOP）
    - ajax（异步请求获取数据）
    - jquery（页面脚本动作+页面动态表格操作）
    - mysql（数据库基本操作）
    - SSH整合
    - 反射（反射的应用）
    - Maven构建项目
  + **项目追加知识点**

1. 反射泛型类型（制作通用类）
2. 视图值与真实值应用
3. 报表工具（java读写Excel、jfreechart工具）
4. Spring计时器任务
5. Spring整合JavaMail（邮件发送）
6. jquery（jquery-struts2-json整合）
   * **项目实用技巧**
7. 通用类抽取制作
8. 代码自动生成器
9. long型日期时间处理+页面日期组件
10. 统一的异常处理方案
11. 全局国际化应用
12. 页面动态表现
13. 遮罩层应用
14. 权限校验系统
15. 通用分页组件制作与使用（自定义标签）
16. 动态数据库表结构

* 项目介绍

ERP：http://baike.baidu.com/subview/109408/7177679.htm?from\_id=22997&type=syn&fromtitle=ERP&fr=aladdin

案例演示

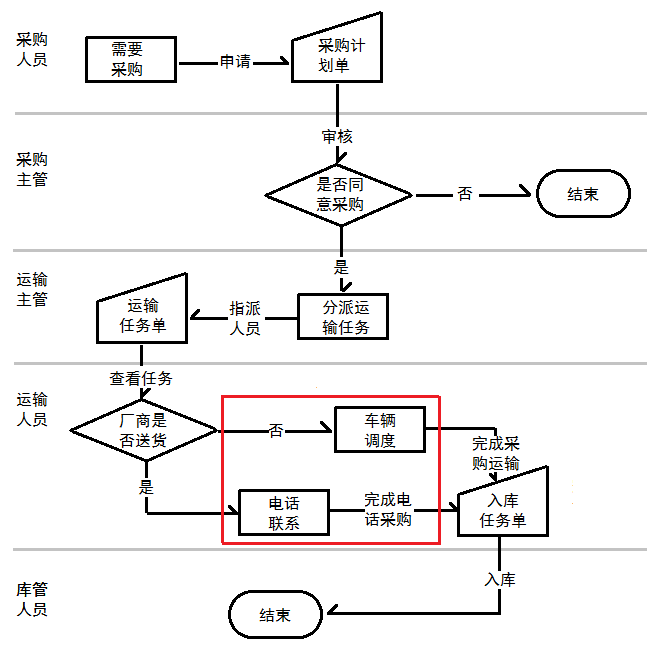
* 主线流程分析

前置模块：主线流程制作需要满足以下单元模块制作完成为基础条件：

员工管理、权限校验系统、供应商管理、供应商商品列别管理、商品管理

采购流程

下采购单→采购审批→判定供应商是否送货→转入运输部门→运输任务指派→运输任务完成→按采购订单任务入库→流程结束



销售流程

下销售单→销售审批→判定库存是否足够→转入分支采购流程→判定商家是否上门提货→转入运输部门→运输任务指派→按销售订单任务出库→运输任务完成→流程结束

（图略）

采购退货流程

下采购退货单→采购退货审批→判定商家是否上门提货→转入运输部门→运输任务指派→按采购退货单任务出库→运输任务完成→流程结束

（图略）

销售退货流程

下销售退货单→销售退货审批→判定商家是否上门送货→转入运输部门→运输任务指派→运输任务完成→按销售退货单任务入库→流程结束

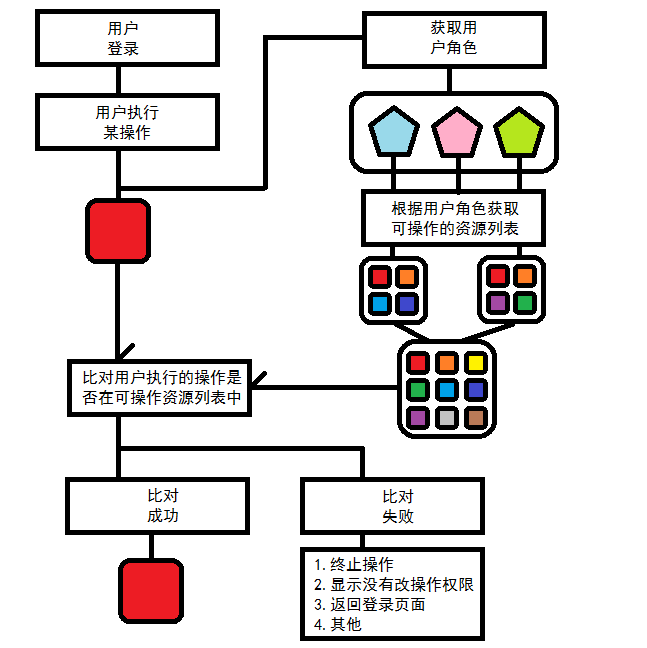
（图略）

* 子线流程分析

权限管理

权限管理是所有管理类系统中必不可少的一部分，权限管理通过数据的设置完成整体系统的操作能力控制，避免用户进行非法操作。

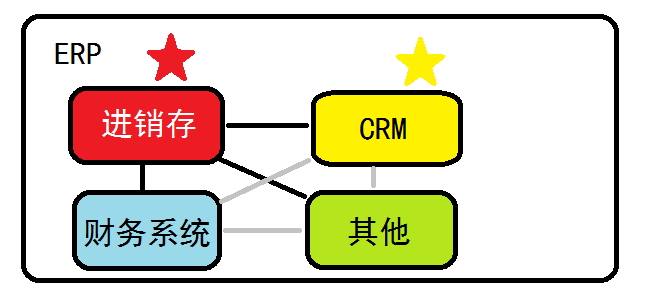
本项目中权限管理要进行具体的实现



CRM项目对接

CRM（Customer Relationship Management）即客户管理系统，用于对客户信息，客户关系进行维护。CRM是一套独立的系统，各种大企业均拥有自主的客户关系管理系统。使用该系统对老客户进行关系维护，对新客户的发展进行实时跟踪，保持良好的发展态势。

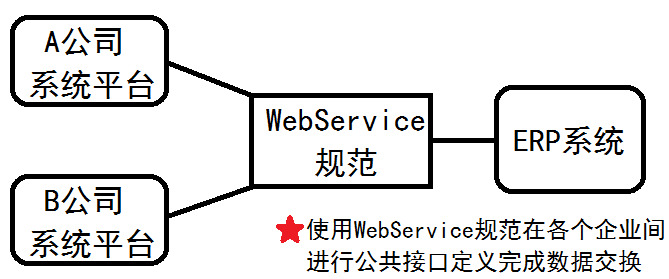
本项目中采购对应的供应商和销售对应的客户实际制作均采用CRM项目进行管理。



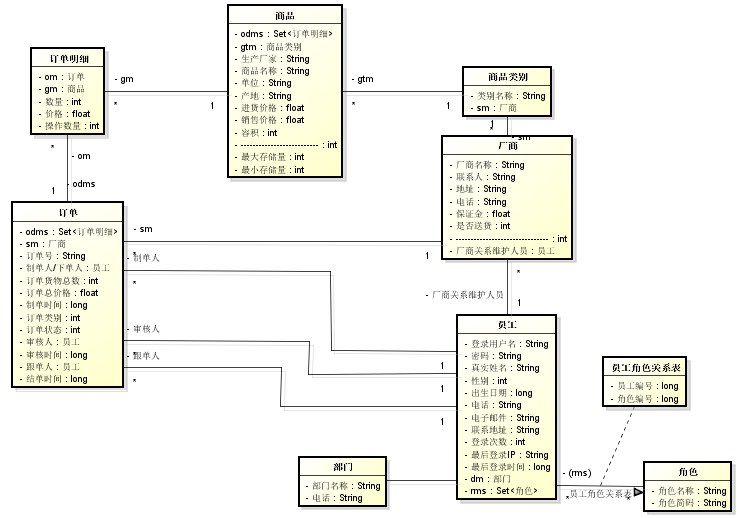
WebService接口对接

企业级应用中，每个企业的管理系统均采用独立控制，独立管理的方式进行。由于各个公司间系统存在有差异，不能进行良好的数据交互。例如A公司给B公司下了一份采购订单，但是B公司并不知情，在B公司的管理系统中也无法即时呈现对应的数据，造成管理系统只为企业管理完成了一半的任务，没有真正意义上的将数据流进行到底。基于此现象，为B公司制作对外的服务接口，A公司给B公司下订单的同时，调用B公司的对外服务接口，这样A公司内部的数据管理维护为正确的，同时B公司也接到的响应的信息，只需要将对外接口进行一定的安全管理就可以轻松实现两个系统间数据交互的模型。

本项目中不涉及WebService接口的适用，希望学员在对应课程学习完毕后，将对应知识添加到项目介绍与简历制作过程中。



* 项目类图分析



* 项目如何服务于企业面试

项目在简历中的描述方式

LYMS是为蓝源公司定制的一套企业ERP系统，包含供应链管理、销售与市场管理、物流控制管理、财务管理、基础平台五个子系统，下辖分销、客户服务、财务管理、库存管理、人力资源、报表、工作流服务、企业信息系统、质量管理、运输管理、项目管理、法规与标准和过程控制等四十多个功能。

本人参与设计并实现了LYMS中的权限控制、采购与采购退货管理、销售与销售退货管理、运输管理、仓库出入库管理及相关功能的报表制作。

项目在面试中描述的几点注意

先从项目总体进行概括性描述，然后介绍4个主线任务，重点突出项目的业务流程，而不是实现技术，尽量回避技术实现方式，以介绍业务为主。从四个核心业务中抽取一个业务为例，给面试官呈现业务流程的复杂度及个人思路的清晰程度。使面试官第一反应认为该项目你全程参与了，并且进行了相应的设计与开发。

由于该项目涉及内容过多，需要学员在下面课程的学习中，将后期所学知识与本项目联系起来，例如WebService和工作流。在这些知识学习完毕后，需要融会贯通到此项目中。这样才能做到不被面试官在技术层面问倒。

关于实现方式方面的问题，面试时不要强调细节，会给对方一种露怯的感觉，不知道说什么，只能拿技术实现方式向项目上堆砌，给人一种只做了这个功能的印象，应该尽量以大的方面来描述。如果被面试官问到具体细节，剖析需求，而不是实现方式。除非被问到如何实现，才从技术层面回应对方，在回应时一步到位，将所用的每个技术完成的任务是什么，为什么用这个技术描述给对方，让对方相信你的选型是经过考虑的，而不是只会用这一个技术，对于不能进行技术间的优缺点比对的学员，以项目经理技术选型为该技术为理由，将问题抛出。

尽量不要总结通过该项目收获了什么，这样会给人一种初入行的感觉，本套ERP系统不适合描述为新手所能完成的任务。该项目的进入，基本上奠定了你的开发能力与开发经验至少在两到三年，如果描述收获，给人一种新手上路的感觉。

把这个项目支持起来需要的经验不低于两年的开发经验，切勿犯低级错误，让人感觉话语间不是一个有经验的开发人员。

本项目中没有太多过新的技术，该项目的亮点在于业务流程的复杂度，沉稳应对才是上策。由于项目中业务过于复杂，能不回答的问题尽量不要主动向外抛出，避免给自己面试带来不必要的麻烦。

* 项目计划

|  |  |  |
| --- | --- | --- |
| 周期 | 任务 | 具体内容 |
| 第一天 | 项目需求分析  Maven环境搭建  登录  未登录用户校验 | 项目介绍  主线流程分析  项目类图分析  项目如何服务于企业面试  项目构建：Maven环境搭建（操作）  静态页面导入（操作）  登录功能（操作）  未登陆用户校验（理解+操作） |
| 第二天 | 部门管理  员工管理 | 部门管理（操作）  遮罩层应用（操作Q）  统一的异常处理方式（理解+操作）  国际化标准制作方案（操作）  按条件查询通用设计（理解+操作）  通用分页（操作Q）  通用基类抽取（理解）  员工管理（操作）  代码生成器制作（操作） |
| 第三天 | 权限系统分析设计实现 | 权限模型设计分析（理解）  资源模型实现（操作Q）  角色模型实现（操作Q）  员工管理二次开发（操作Q）  权限设置（操作）  权限访问过滤（理解+操作） |
| 第四天 | 权限树制作 | 权限树模型分析（操作）  权限树制作（操作Q） |
| 第五天 | 供应商管理  商品类别管理  商品信息管理 | 供应商维护（操作）  商品类别维护（操作）  商品信息维护（理解+操作Q） |
| 第六天 | 采购申请 | 采购流程复习（理解）  状态字段设计（理解）  订单生成（操作QQQ） |
| 第七天 | 采购审核、商品运输 | 采购审核（理解+操作）  商品运输任务指派（理解+操作）  商品运输任务完成（理解+操作） |
| 第八天 | 仓库管理  入库 | 仓库管理（操作）  订单商品入库（理解+操作QQQ）  仓库商品明细管理（操作）  仓库出入库日志管理（操作） |
| 第九天 | 报表 | 报表设计思想（理解）  报表制作（理解+操作QQQ）  Jfreechart（操作Q）  jxl（操作） |
| 第十天 | 项目总结 | Spring定时器（操作）  常用数据频度维护（操作）  库存预警功能（操作）  JavaMail（操作）  项目总结 |

## 广西移动电商

### 1.说说你最近做的这个项目的背景？

随着中国通信行业竞争程度的加剧，竞争的形态也发生了巨大的变化，从以产品、价格为主的竞争转向以服务为主的竞争，服务成为主导竞争格局的重要因素。渠道作为企业完成客户沟通、产品/服务交换过程以及实现价值、产生效益的重要载体，发挥了采集、传达客户和竞争对手等市场信息，为买卖双方提供便利，协调供需矛盾，为客户提供合适的产品与服务，向客户传递产品/服务信息，实现营销/服务目标等重要的功能。

广西移动上线移动商城，一方面可以带动传统业务绩效提升，增强客户满意度和粘性，另一方面，也为基于互联网的商务模式创新奠定基础。

针对上述行业环境变化和业务战略目标，广西移动在网上终端预约销售基础上，即将启动网上商城建设项目，用于建立网上终端、营销案在线销售及相关辅助功能，包含商品管理、订单管理、类目管理、客户管理、合作商管理、客服管理、购物平台、内容管理等，很大程度上分担了人工的压力，对提高客户服务效率和客户满意度能够起到较好的作用。

基于此，广西移动提出建设网上商城建设项目工程。

**广西移动是按省进行管理(那么写的时候可以按各个省进行说)**

**电商的两种运营模式：**

**B2C：商家对用户**

**B2B：商家对商家，再对用户**

**手机、营销案(移动BOSS端)、卡号三种商品！此外还有团购和秒杀活动(需要系统性能比较好来进行支撑)**

**除了内部系统，一般的系统都有边界，它都会与其他的项目有交叉。电子渠道(电商项目与其他项目的合集！)**



**Boss系统是电商商城的核心，所有的营销案都在这里面。所有的功能交给统一的管理平台，这里需要具备单点登陆的功能。**

**CMS内容管理系统，主要做系统的页面静态化管理，文字、关键字、敏感词的管理，这里归并到了后台管理里面了。**

### 2.整个项目的功能架构如何？



**系统前台**

1. **单品页面(终端+营销案)**
2. **购物车(面试经常被问到)**
3. **首页**
4. **用户中心**

**系统后台 【对账管理：每天都要进行下单与银行的钱是否一致】**

1. **订单管理：使用activiti工作流进行处理**
2. **静态发布：**
3. **商品管理**

### 3.这个项目为用户提供了哪些服务？包括哪些模块？

网上商城项目，用于建立网上终端、营销案和号卡在线销售及相关辅助功能，后台包含商品管理、订单管理、类目管理、客户管理、合作商管理、客服管理、支付平台、内容管理等，很大程度上分担了人工的压力，前台包括个人中心，购物车，商城首页，频道页，商品详情页（静态化），提交订单页，支付页面等页面构成，对提高客户服务效率和客户满意度能够起到较好的作用。

### 4.你承担这个项目的哪些核心模块

* + - 商品管理模块：其中包括品牌管理，属性管理商品录入/上下架管理，商品添加审核，静态页面发布
    - 订单模块：其中包括使用activiti工作流订单的查询和订单的流转,定时作废
    - 商品前台首页：其中主要负责首页商品列表筛选，首页上动态展示筛选条件，点击每一个筛选条件下面的商品列表要做联动
    - 单品页面：采用freemarker来实现页面静态化，展示商品详情信息和商品购买，该页面采用静态化以减轻系统压力，使用了cxf框架发布服务
    - 提交订单页面：提交用户的订单信息, 处理并发问题。
    - 个人中心，包括用户的登录，个人信息的管理，收货地址的管理，用户所下的订单的管理
    - 购物车：把购物车的信息存在cookie里面管理

### 5.这些模块的实现思路说一下

**①商品管理模块**

品牌管理：

主要掌握的核心是品牌的添加

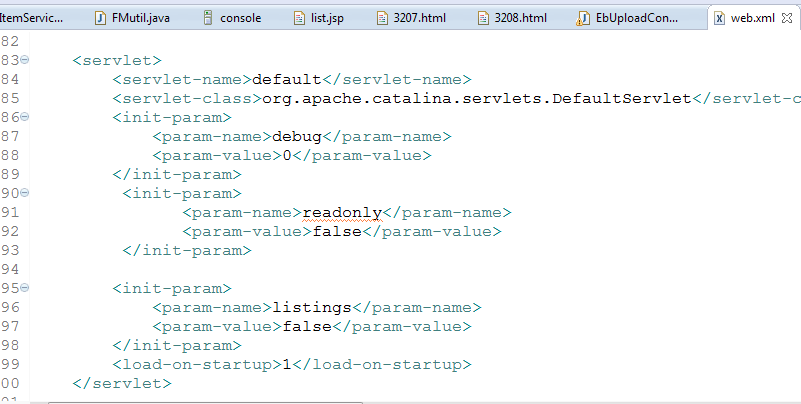
* 图片服务器的搭建，
* 上传图片到图片服务器
* 表单的验证，离开焦点的验证，点击完成时的验证，后台服务器的ajax验证，表单规范
* 防止表单的二次提交
* 编辑时不需要修改品牌名，区别readOnly和disabled
* 删除时的二次确认

注意：

自定义属性必须掌握：html中自定义的属性可以帮助索引元素

**图片服务器的搭建**

1. 创建一个maven的web工程，在工程中创建一个存放资源的目录
2. 把tomcat的web.xml中DefaultServlet的只读属性改成false



1. 编写上传到图片服务器的代码

由于应用服务器与图片服务器出于不同的两台机器之中，所以可以提高系统的性能，能起到负载均衡的作用。上传图片时使用**J**ersey 客户端 API 调用 REST 风格的 Web 服务,  Jersey 1 是一个开源的、可以用于生产环境的 JAX-RS（RESTful Web Services 的 Java API 规范，JSR-311）实现。通过 Jersey 可以很方便的使用 Java 来创建一个 RESTful Web Services。

**byte**[] fileByte = commFile.getBytes();

//创建客服端，基于webservice

Client client = Client.*create*();

//指定资源路径

WebResource webResource = client.resource(Constants.*picPath*+fileName);

//使用put的请求方式把资源文件放到资源服务器上

webResource.put(String.**class**, fileByte);

前台使用ajax提交表单，需要使用jquery的*jquery.form.js*插件

$("#form").ajaxSubmit({

url:url,

type:"post",

dataType:"text",

data:{

...

},

//beforeSubmit:validate,

success:**function**(responseText){

**var** obj = $.parseJSON(responseText);

},

error:**function**(){

}

});

1. 图片服务器中Upload文件夹中随便建立一个文件，防止空文件夹的情况下发布后upload消失

**表单验证**

1. 提交时做验证

做好约定，每个文本中设置reg属性和tip自定义的属性，reg存放正则表达式，tip中存放不合法时的提示信息，还有品牌名称重复的验证。

reg2,tip属于自定义的属性，这种定义方式方便使用jquery的属性选择器

<p>

<label><samp>\*</samp>品牌名称：</label>

<input type="text" id="brandName" name="brandName" class="text state" reg2="^[a-zA-Z0-9\u4e00-\u9fa5]{1,20}$" tip="必须是中英文或数字字符，长度1-20"/>

<span></span>

</p>

1. 在表单提交时做验证使用$(“form”).submit(function(){ return false });,必填字段和非必填的字段需要区别对待

$("#form111").submit(**function**(){

**var** isSubmit = **true**;

$(**this**).find("[reg2]").each(**function**(){

**var** regStr = $(**this**).attr("reg2");

//剪掉值中的两侧的字符串

**var** value = $.trim($(**this**).val());

**var** tip = $(**this**).attr("tip");

//创建正则表达式的对象

**var** reg = **new** RegExp(regStr);

**if**(!reg.test(value)){

$(**this**).next("span").html(tip);

isSubmit = **false**;

//跳出循环,在jquery的each语句之中跳出循环使用return false; 如果在原生js里面可使用break;, return;:代表终止执行程序

**return** **false**;

}

});

$(**this**).find("[reg1]").each(**function**(){

**var** regStr = $(**this**).attr("reg1");

**var** value = $.trim($(**this**).val());

**var** tip = $(**this**).attr("tip");

**var** reg = **new** RegExp(regStr);

**if**(value != **null** && value != ""){

**if**(!reg.test(value)){

$(**this**).next("span").html(tip);

isSubmit = **false**;

**return** **false**;

}

}

});

**return** isSubmit;

});

1. 使用离焦事件做友好的提示

$("input[reg2]").blur(**function**(){

**var** regStr = $(**this**).attr("reg2");

**var** value = $.trim($(**this**).val());

**var** tip = $(**this**).attr("tip");

**var** reg = **new** RegExp(regStr);

**if**(!reg.test(value)){

$(**this**).next("span").html(tip);

}**else**{

$(**this**).next("span").html("");

}

});

1. 表单的二次提交处理

* 锁屏
* 锁按钮

**②商品的查询**

商品查询需要组合条件加分页查询

* 组合条件：品牌，审核状态，商品名称，需要动态sql

<select id=*"queryItemByCondtion"* resultMap=*"BaseResultMap"* parameterType=*"map"*>

select \*

from (select a.\*, rownum rm

from (

select \*

from eb\_item ei

<where>

<if test=*"brandId != null"*>

ei.brand\_id = #{brandId}

</if>

<if test=*"auditStatus != null"*>

and ei.audit\_status = #{auditStatus}

</if>

<if test=*"showStatus != null"*>

and ei.show\_status = #{showStatus}

</if>

<if test=*"itemName != null"*>

and ei.item\_name like '%${itemName}%'

</if>

</where>

order by ei.item\_id desc) a

<![CDATA[

where rownum < #{endNum}) b

where b.rm > #{startNum}

]]>

</select>

*--查询大于当前页首行号，主要解决oracle的rownum不支持大于号的问题*

select \*

from (

*--查询小于当前最大行号的数据*

select rownum rm,a.\*

from (

*--第一个select查询所有的业务数据*

select \* from eb\_item

) a

where rownum < 21) b

where b.rm > 10

* 分页查询

1. 查询结果集的sql，传入开始行数和结束的行数

select \*

from (select a.\*, rownum rm

from (

...

内部sql

...

) a where rownum < #{endNum}) b

where b.rm > #{startNum}

1. 使用内部sql查询结果集的总条数
2. 使用分页工具类，创建page对象更换每次的数据总条数和当前页数和每页的条数，查询出结果集后把结果集注入到page对象之中

public class Page {

int totalCount = 0;

int pageSize = 10;

int currentPageNo = 1;

int startNum = 0;

int endNum = 11;

public Page(int totalCount, int pageSize, int currentPageNo) {

super();

this.totalCount = totalCount;

this.pageSize = pageSize;

this.currentPageNo = currentPageNo;

}

public int getStartNum(){

return (currentPageNo - 1) \* pageSize;

}

public int getEndNum(){

return currentPageNo \* pageSize + 1;

}

public int getTotalPage(){

int totalPage = totalCount/pageSize;

if(totalPage == 0 || totalCount%pageSize != 0){

totalPage ++;

}

return totalPage;

}

public int getNextPage(){

if(currentPageNo >= getTotalPage()){

return currentPageNo;

}else{

return currentPageNo + 1;

}

}

public int getPrePage(){

if(currentPageNo <= 1){

return currentPageNo;

}else{

return currentPageNo - 1;

}

}

List<?> list;

public List<?> getList() {

return list;

}

public void setList(List<?> list) {

this.list = list;

}

public int getTotalCount() {

return totalCount;

}

public void setTotalCount(int totalCount) {

this.totalCount = totalCount;

}

public int getPageSize() {

return pageSize;

}

public void setPageSize(int pageSize) {

this.pageSize = pageSize;

}

public int getCurrentPageNo() {

return currentPageNo;

}

public void setCurrentPageNo(int currentPageNo) {

this.currentPageNo = currentPageNo;

}

public void setStartNum(int startNum) {

this.startNum = startNum;

}

public void setEndNum(int endNum) {

this.endNum = endNum;

}

}

4． 制作前端样式

**var** currentPageNo = parseInt($("#currentPageNo").val());

**var** totalCount = parseInt($("#totalCount").val());

**var** totalPage = parseInt($("#totalPage").val());

$("#pagePiece").html(totalCount);

$("#pageTotal").html(currentPageNo+"/"+totalPage);

**if**(currentPageNo <= 1){

$("#previous").hide();

}**else**{

$("#previous").show();

}

**if**(currentPageNo >= totalPage){

$("#next").hide();

}**else**{

$("#next").show();

}

$("#next").click(**function**(){

$("#pageNo").val(parseInt(currentPageNo)+1);

$("#form1").submit();

});

$("#previous").click(**function**(){

$("#pageNo").val(parseInt(currentPageNo)-1);

$("#form1").submit();

});

Oracle 分页sql的描述：

1.最内层的写查询当前表的全量即可

2.对于oracle数据库分页需要依赖于rownum，但是rownum不支持大于号，但是支持小于号，可以rownum小于结束行号查询出来一个结果集（在全量的外层套一个select，它的结果集需要把rownum作为结果返回）

3.在第二步的结果集基础上再做一次查询，查询条件以第二步查询出来的rownum的值作为条件大于开始行号即可

**③商品发布**

Console和portal是分开部署在两台服务器上，发布需要在console端去控制，但是生成的静态化的文件要发布到portal的工程之中，所以发布的服务要在portal上，但是要在console中来调用，异构之间的调用要使用webservice。

1. 采用cxf的webservice框架来整合spring
2. 在web.xml中来配置cxf的核心servlet

<servlet>

<servlet-name>cxfServlet</servlet-name>

<servlet-class>org.apache.cxf.transport.servlet.CXFServlet</servlet-class>

<!-- <init-param>

<param-name>config-location</param-name>

<param-value>classpath:cxf-servlet.xml</param-value>

</init-param>

<load-on-startup>1</load-on-startup> -->

</servlet>

<servlet-mapping>

<servlet-name>cxfServlet</servlet-name>

<url-pattern>/services/\*</url-pattern>

</servlet-mapping>

1. 创建服务的接口和接口的实现类，注意接口上加上@WebService注解
2. 创建cxf的核心配置文件cxf-servlet.xml,配置带有接口的webservice服务使用

<jaxws：server>标签

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

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

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xmlns:jaxws=*"http://cxf.apache.org/jaxws"*

xmlns:jaxrs=*"http://cxf.apache.org/jaxrs"* xmlns:cxf=*"http://cxf.apache.org/core"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans*

*http://www.springframework.org/schema/beans/spring-beans.xsd*

*http://cxf.apache.org/jaxrs http://cxf.apache.org/schemas/jaxrs.xsd*

*http://cxf.apache.org/jaxws http://cxf.apache.org/schemas/jaxws.xsd*

*http://cxf.apache.org/core http://cxf.apache.org/schemas/core.xsd"*>

<!-- 引入CXF Bean定义如下,早期的版本中使用 -->

<import resource=*"classpath:META-INF/cxf/cxf.xml"* />

<import resource=*"classpath:META-INF/cxf/cxf-extension-soap.xml"* />

<import resource=*"classpath:META-INF/cxf/cxf-servlet.xml"* />

<jaxws:server id=*"publish"* address=*"/publish"* serviceClass=*"cn.itcast.ecps.ws.service.EbItemWSService"*>

<jaxws:serviceBean>

<bean class=*"cn.itcast.ecps.ws.service.impl.EbItemWSServiceImpl"*></bean>

</jaxws:serviceBean>

<!-- 输入输出的拦截器 -->

<jaxws:inInterceptors>

<bean class=*"org.apache.cxf.interceptor.LoggingInInterceptor"*></bean>

</jaxws:inInterceptors>

<jaxws:outInterceptors>

<bean class=*"org.apache.cxf.interceptor.LoggingOutInterceptor"*></bean>

</jaxws:outInterceptors>

</jaxws:server>

</beans>

1. 修改cxf-servlet.xml的位置，在spring的listener中加载

<listener>

<listener-class>

org.springframework.web.context.ContextLoaderListener

</listener-class>

</listener>

<context-param>

<param-name>contextConfigLocation</param-name>

<param-value>classpath\*:beans.xml,classpath\*:cxf-servlet.xml

</param-value>

</context-param>

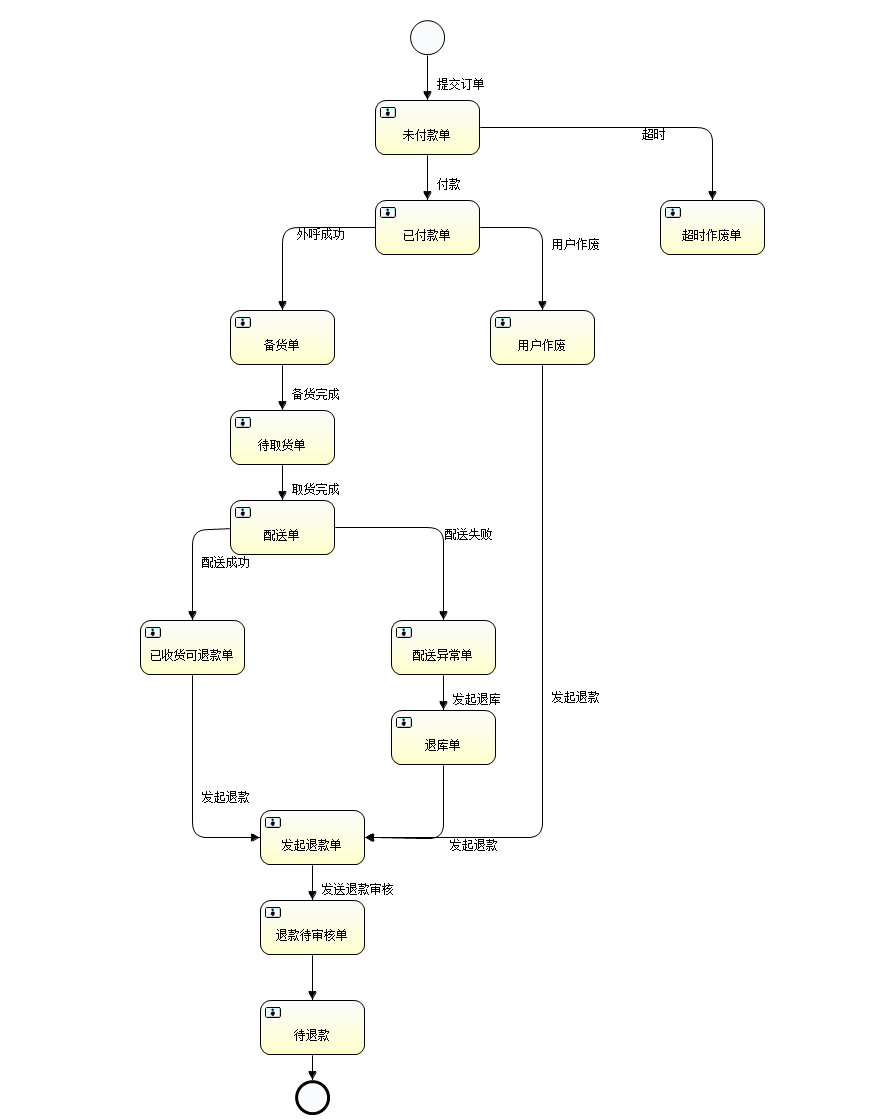
1. 启动服务器发布webservice的服务，使用wsdl2java生成客户端的代码

Wsdl2java –d . –p cn.itcast.ecps.ws.stub [http://.........wsdl](http://.........wsdl/)?

1. 在客户端调用。

**④订单管理模块**

1) 流程设计



2) 订单整合activiti工作流

1.activiti流程和spring整合

创建activiti-context.xml文件

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

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

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*

*http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-2.5.xsd*

*http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-3.0.xsd"*>

<bean id=*"processEngineConfiguration"* class=*"org.activiti.spring.SpringProcessEngineConfiguration"*>

<!-- 数据源 -->

<property name=*"dataSource"* ref=*"dataSource"* />

<!-- 配置事务管理器，统一事务 -->

<property name=*"transactionManager"* ref=*"txManager"* />

<!-- 设置建表策略 -->

<property name=*"databaseSchemaUpdate"* value=*"true"* />

</bean>

<bean id=*"processEngine"* class=*"org.activiti.spring.ProcessEngineFactoryBean"*>

<property name=*"processEngineConfiguration"* ref=*"processEngineConfiguration"* />

</bean>

<bean id=*"repositoryService"* factory-bean=*"processEngine"* factory-method=*"getRepositoryService"* />

<bean id=*"runtimeService"* factory-bean=*"processEngine"* factory-method=*"getRuntimeService"* />

<bean id=*"taskService"* factory-bean=*"processEngine"* factory-method=*"getTaskService"* />

<bean id=*"historyService"* factory-bean=*"processEngine"* factory-method=*"getHistoryService"* />

</beans>

1. 画流程图
2. 创建流程服务类

package cn.itcast.service.impl;

import java.io.File;

import org.activiti.engine.RepositoryService;

import org.activiti.engine.repository.DeploymentBuilder;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import cn.itcast.service.IWorkFlowService;

//@Service

public class WorkflowServiceImpl implements IWorkFlowService {

@Autowired

RepositoryService repositoryService;

/\*public RepositoryService getRepositoryService() {

return repositoryService;

}

public void setRepositoryService(RepositoryService repositoryService) {

this.repositoryService = repositoryService;

}\*/

public void deployFlow() {

//创建发布流程配置对象

DeploymentBuilder builder = repositoryService.createDeployment();

//指定流程资源路径

builder.addClasspathResource("activit-orderflow.bpmn")

.addClasspathResource("activit-orderflow.png");

builder.deploy();

}

}

1. 部署流程
2. 查询业务任务
3. 办理任务

### 6.项目中哪些功能模块涉及了大数据量访问，怎么解决

系统前台是互联网上的用户访问的，会有大量用户来访问。

**假定有1w个人打开你的网站来订商品，问你如何解决并发问题(可扩展到任何高并发网站要考虑的并发读写问题)**

 问题，1w个人来访问，商品没出去前要保证大家都能看到有商品，不可能一个人在看到商品的时候别人就不能看了。到底谁能抢到，那得看这个人的“运气”（网络快慢等）

 其次考虑的问题，并发，1w个人同时点击购买，到底谁能成交？总共只有一张商品。

Update eb\_sku t sett.stock = t.stock – 1 where t.sku\_id = #{skuId} and t.stock > 0

Update eb\_sku t set t.sale = t.sale +1 where t.sku\_id = #{skuId}

 首先我们容易想到和并发相关的几个方案 ： 锁 同步

 同步更多指的是应用程序的层面，多个线程进来，只能一个一个的访问，java中指的是syncrinized关键字。锁也有2个层面，一个是java中谈到的对象锁，用于线程同步；另外一个层面是数据库的锁；如果是分布式的系统，显然只能利用数据库端的锁来实现。

DB（锁要加在数据库上）

war锁（所有的用户排队）

War

假定我们采用了同步机制或者数据库物理锁机制，如何保证1w个人还能同时看到有商品，显然会牺牲性能，在高并发网站中是不可取的。使用hibernate后我们提出了另外一个概念：**乐观锁（一定要用）**、**悲观锁**（即传统的物理锁）；采用乐观锁即可解决此问题。乐观锁意思是不锁定表的情况下，利用业务的控制来解决并发问题，这样即保证数据的并发可读性又保证保存数据的排他性，保证性能的同时解决了并发带来的脏数据问题。

 hibernate中如何实现乐观锁：

 前提：在现有表当中增加一个冗余字段，version版本号, long类型  
 原理：

1）只有当前版本号》=数据库表版本号，才能提交  
 2）提交成功后，版本号version ++

 实现很简单：在ormapping增加一属性**-lock="version"即可，以下是样例片段optimistic**

<hibernate-mapping>  
 <class name="com.insigma.stock.ABC" **optimistic-lock="version"** table="T\_Stock" schema="STOCK">

更新的时候给版本号字段加上 1，然后 UPDATE 会返回一个更新结果的行数，通过这个行数去判断。

|  |  |  |
| --- | --- | --- |
| Sku\_id | sale | Version(乐观锁的标志字段) |
| 1 | 100 | 1 |

100

100  
UPDATE 必须这样写：

SQL code

[?](http://bbs.csdn.net/topics/390339713?page=1" \l "clipboardWindow" \o "源码)

|  |  |
| --- | --- |
| 1  2  3  4  5 | Sale = 100  UPDATE EB\_SKU u     SET u.SALES = #SALES#,         u.version = u.version + 1   WHERE u.SKU\_ID = #SKUID#     AND u.version = #version# |

Update user t set t.address = #{address} where t.user\_id = #{userId}  
如果更新执行返回的数量是 0 表示产生并发修改了，需要重新获得最新的数据后再进行更新操作。  
Hibernate、JPA 等 ORM 框架或者实现，是使用版本号，再判断 UPDATE 后返回的数值，如果这个值小于 1 时则抛出乐观锁并发修改异常。

**2，大数据量你是如何考虑的**

 再比如，中国移动有上亿的用户量，表如何设计？把所有用于存在于一个表么？

 所以，大数量的系统，必须考虑表拆分-（表名字不一样，但是结构完全一样），通用的几种方式：（视情况而定）

1）按业务分，比如 手机号的表，我们可以考虑 130开头的作为一个表，131开头的另外一张表 以此类推

userXXX

map(“tablename”, “user\_130”)

Select \* from #{user\_130} t where t.phone = #{phone}

 2）利用oracle的表拆分机制做分表

3）如果是交易系统，我们可以考虑按时间轴拆分，当日数据一个表，历史数据弄到其它表。这里历史数据的报表和查询不会影响当日交易。

seleCash\_2014-5-14

当然，表拆分后我们的应用得做相应的适配。单纯的or-mapping也许就得改动了。比如部分业务得通过存储过程等

===============================================================================

1. 解决大量用户访问量问题方案是集群部署，防止宕机， 负载均衡
2. 我们采用4台portal服务器来集群，使用2台nginx代理服务器，
3. 反向代理，把4台portal的服务（host）集中起来，访问动态链接代理地址（代理IP：192.168.1.100）时候会把请求转发到4太portal上，如果访问静态页面直接访问nginx上的资源文件就可以了，静态html中有ajax的请求由nginx的反向代理功能来转发。
4. 部署静态资源（html和图片）
5. Rsync用作资源同步

当console上传的图片，由于rsync的部署，会指定一个具体的同步目录（上传图片的目录），一旦发现目录中有文件就立刻同步到nginx上

1. Redis负责管理session和缓存搜索的数据

管理session的原因：用于多台服务器之间需要有相同的session，同享策略耗费资源，所以采用redis来存储session。

缓存频繁被搜索的数据。

### 7.你碰到了哪些问题？你是怎么解决的？

1. 开发webservice接口出现客户端和服务端不同步，导致接口无法测试，产生的原因沟通不畅。
2. 订单提交时由于本地bug或者意外故障导致用户钱支付了但是订单不成功，采用对账方式来解决。
3. 上线的时候一定要把支付的假接口换成真接口。
4. 项目中用到了曾经没有用过的技术，解决方式：用自己的私人时间主动学习

5.在开发过程中与测试人员产生一些问题，本地环境ok但是测试环境有问题，环境的问题产生的，浏览器环境差异，服务器之间的差异

6.系统运行环境问题，有些问题是在开发环境下OK，但是到了测试环境就问题，比如说系统文件路径问题、导出报表中的中文问题（报表采用highcharts），需要在系统jdk中添加相应的中文字体才能解决；

### 8.做完这个项目有哪些收获？

首先，在数据库方面，我现在是真正地体会到数据库的设计真的是一个程序或软件设计的重要和根基。因为数据库怎么设计，直接影响到一个程序或软件的功能的实现方法、性能和维护。由于我做的模块是要对数据库的数据进行计算和操作的，所以我对数据库的设计对程序的影响是深有体会，就是因为我们的数据库设计得不好，搞得我在对数据库中的数据进行获取和计算利润、总金时，非常困难，而且运行效率低，时间和空间的复杂也高，而且维护起来很困难，过了不久，即使自己有注释，但是也要认真地看自己的代码才能明白自己当初的想法和做法。加上师兄的解说，让我对数据库的重要的认识更深一层，数据库的设计真的是重中之重。

其次，就是分工的问题。虽然这次的项目我们没有在四人选出一个组长，但是，由于我跟其他人都比较熟，也有他们的号码，然后我就像一个小组长一样，也是我对他们进行了分工。俗话也说，分工合作，分好了工，才能合作。但是这次项目，我们的分工却非常糟糕，我们在分工之前分好了模块，每个模块实现什么功能，每个人负责哪些模块。本以为我们的分工是明确的，后来才发现，我们的分工是那么的一踏糊涂，一些功能上紧密相连的模块分给了两个人来完成，使两个人都感到迷惘，不知道自己要做什么，因为两个人做的东西差不多。我做的，他也在做，那我是否要继续做下去？总是有这样的疑问。从而导致了重复工作，浪费时间和精力，并打击了队员的激情，因为自己辛辛苦苦写的代码，最后可能没有派上用场。我也知道，没有一点经验的我犯这样的错是在所难免，我也不过多地怪责自己，吸取这次的教训就好。分工也是一门学问。

再者，就是命名规范的问题。可能我们以前都是自己一个人在写代码，写的代码都是给自己看的，所以我们都没有注意到这个问题。就像师兄说的那样，我们的代码看上去很上难看很不舒服，也不知道我们的变量是什么类型的，也不知道是要来做什么的。但是我觉得我们这一组人的代码都写得比较好看，每个人的代码都有注释和分隔，就是没有一个统一的规范，每个人都人自己的一个命名规则和习惯，也不能见名知义。还有就是没有定义好一些公共的部分，使每个人都有一个自己的“公共部分”，从而在拼起来时，第一件事，就是改名字。而这些都应该是在项目一开始，还没开始写代码时应该做的。

然后，我自己在计算时，竟然太大意算错了利润，这不能只一句我不小心就敷衍过去，也是我的责任，而且这也是我们的项目的核心部分，以后在做完一个模块后，一定要测试多次，不能过于随便地用一个数据测试一下，能成功就算了，要用可能出现的所有情况去测试程序，让所有的代码都有运行过一次，确认无误。

最后，也是我比较喜欢的东西，就是大家一起为了一个问题去讨论和去交流。因为我觉得，无论是谁，他能想的东西都是有限的，别人总会想到一些自己想不到的地方。跟他人讨论和交流能知道别人的想法、了解别人是怎样想一个问题的，对于同样的问题自己又是怎样想的，是别人的想法好，还是自己的想法好，好在什么地方。因为我发现问题的能力比较欠缺，所以我也总是喜欢别人问我问题，也喜欢跟别人去讨论一个问题，因为他们帮我发现了我自己没有发现的问题。在这次项目中，我跟植荣的讨论就最多了，很多时候都是不可开交的那种，不过我觉得他总是能够想到很多我想不到的东西，他想的东西也比我深入很多，虽然很多时候我们好像闹得很僵，但是我们还是很要好的! 嘻嘻！而且在以后的学习和做项目的过程中，我们遇到的问题可能会多很多，复杂很多，我们一个人也不能解决，或者是没有想法，但是懂得与他人讨论与交流就不怕这个问题，总有人的想法会给我们带来一片新天地。相信我能做得更好。

还有就是做项目时要抓准客户的要求，不要自以为是，自己觉得这样好，那样好就把客户的需求改变，项目就是项目，就要根据客户的要求来完成。

### 9.项目介绍：

XXX网上商城项目，用于建立网上终端、营销案和号卡在线销售及相关辅助功能，后台包含商品管理、订单管理、类目管理、客户管理、合作商管理、客服管理、支付平台、内容管理等，很大程度上分担了人工的压力，前台包括个人中心，购物车，商城首页，频道页，商品详情页（静态化），提交订单页，支付页面等页面构成，对提高客户服务效率和客户满意度能够起到较好的作用。

xxx商城项目打造的是“社区+电商”的模式，用户不只是在社区中有自己的圈子，还可以将电商加入到社区中，整个电商网站实现的功能非常之多，采用分布式的架构设计，包括后台管理、前台系统、订单系统、单点登录系统、搜索系统、会员系统等。

①该项目是自己公司的产品，我们公司负责整个网站的运营，属于B2C平台；

②系统的用途，主要是提供B2C的平台，其中自营商品也有商家入住，类似天猫。

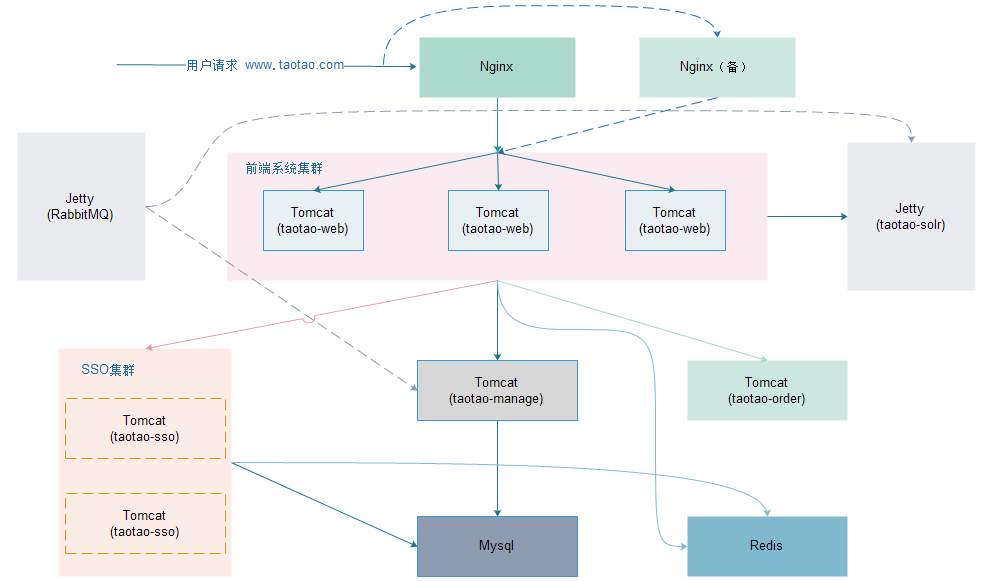
③系统架构，采用分布式的系统架构，其中前台系统和单点登录系统采用了集群的方式部署，在后台管理系统中采用了Maven的多模块化的管理，其中采用了水平切分的方式，将pojo、dao、service、web分层开发，这样做的好处就是可以重用性更高。

系统内部接口调用采用Httpclient，并且使用Httpclient的连接池技术，接口提供端采用RESTful方式的接口定义；

系统之间的通知机制采用MQ的方式，使用RabbitMQ的实现，使用了RabbitMQ的消息订阅模式的消息机制；

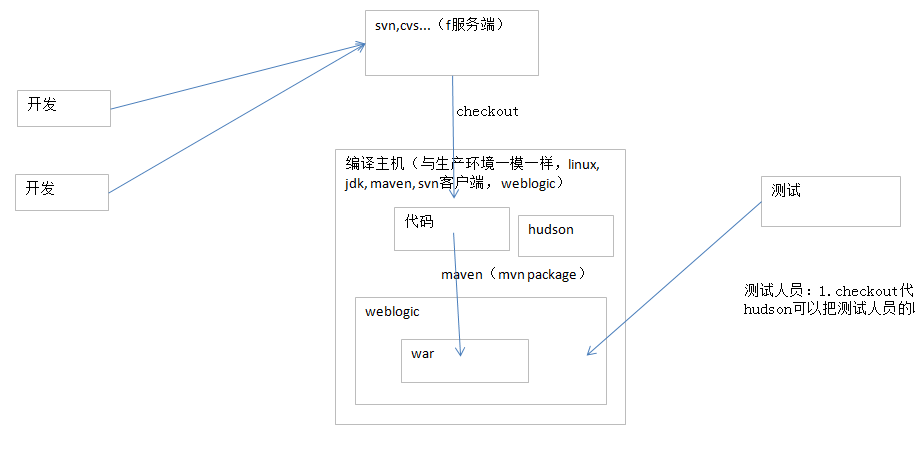
系统的接口还对JS的跨域做了支持，采用了jsonp的解决方法，在后台接口中扩展了spirng提供的jackson数据转化器实现；

④部署方面，采用了Nginx+tomcat的模式，其中nginx的作用一方面是做反向代理、负载均衡、另一方面是做图片等静态资源的服务器。



### 10.介绍项目的步骤

1. 我的项目是XXX系统，该项目是给中国移动做的电商项目，电商项目分为b2b2c,b2b,b2c。
2. 项目的用途：XXX网上商城项目，用于建立网上终端、营销案在线销售及相关辅助功能，在网上来卖裸机， 营销案（合约机），卡号，以及开展一些团购，秒杀一些活动，给省移动公司带来最大的利润。
3. 项目架构：我们的项目是互联网项目，要分成前台后台，前台portal和后台console分别部署在不同主机上，公用一套数据库，导致业务逻辑层和数据访问层代码重复，考虑使用maven项目，前后台一个model，都是web project，再建立公用的业务层model称为core（包含dao），前后台来引用core，这样可以公用一套业务层，提高代码的复用性（使用maven依赖的优点），maven在我们的项目中还有一大优点项目构建，参见下图



1. 项目的模块：商品管理、订单管理、类目管理、客户管理、合作商管理、客服管理、支付平台、内容管理等，很大程度上分担了人工的压力，前台包括个人中心，购物车，商城首页，频道页，商品详情页，提交订单页，支付页面等页面构成
2. 我所负责的模块：商品管理、订单管理 ，购物车 个人中心，商城首页，商品详情页，提交订单页（稍稍展开一点）。

商品管理的设计思路描述:

1. **从大小说(一定举例子)**： 类别（体现自关联）--🡪 品牌（体现和类别的关系）--🡪商品（体现审核，上架）-🡪属性（体现属性和类别的关系）-🡪商品级别的属性-🡪最小销售单元（价钱，库存，销量）--🡪最小销售单元级别的属性
2. 商品流程：商品添加---🡪审核---🡪上架---🡪发布（强调静态化，cxf）

**订单的描述**：

重要的是订单的业务流程的掌握。

**个人中心**：登录，个人资料管理，收货地址管理

**商城的首页：**筛选，把筛选属性展示在页面作为筛选条件，使用iframe来做商品列表加载**页面无刷新：**首页上动态展示筛选条件，点击每一个筛选条件下面的商品列表要做联动。

**商品详情页：**静态化，提高系统的性能，每一个商品一个html页面，数据不需要实时去数据库查。

**购买流程：**注意：判断用户和sku的库存，库存的并发问题。

## 附加内容：

### 1.电商行业技术特点：

①技术新：(NoSql推广首在社区网站和电商项目)，发展快，需求推动技术的革新。

②技术范围广：除了java，像淘宝前端还使用了PHP，数据库MySQL或者oracle，nosql，服务器端使用Linux，服务器安全、系统安全

③分布式：以前是在一台机器上做运算，现在是分散到很多机器上，最后汇总起来。(集中式向分布式进行考虑)由需求来推动

④高并发、集群、负载均衡、高可用：由并发问题采用集群进行处理，其中，集群会涉及服务器的主从以及分布问题，使用负载均衡。(权重高低)高可用是对用户而言，用户的服务不中断(系统升级，服务不中断，淘宝每周更新2次)。

⑤海量数据：双11,570亿的背后，订单有多少？浏览次数有多少？商品会有多少？活动相关数据？

⑥业务复杂：不要简单的认为是：商品展示出来后，加入购物车后购买就完成了。后台特别复杂，比如优惠(包邮、满减)

⑦系统安全：系统上线必须通过系统安全部门审核通过。前年CSDN数据泄露。快捷酒店数据泄露(通过身份证就可以查看你的开房记录)。近几年，安全意识逐步在提高。

### 2.分布式系统架构



### 3.开发的模式



### 4.Maven的垂直划分和水平划分

优缺点：

1. 垂直划分，功能模块明确，层次不清晰，代码重用性查；
2. 水平划分，层次清晰，代码重用性高，独立维护；

淘淘商城的后台管理系统采取水平划分；



聚合工程的打包类型是pom

功能垂直划分



### 5.Nginx的重用命令

start nginx.exe //启动

nginx.exe –s stop //停止

nginx.exe –s reload // 重新加载配置文件

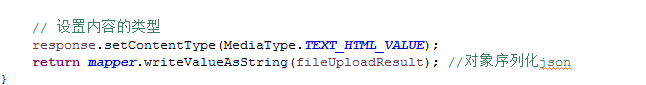
修改配置文件：conf/nginx.conf

### 6.文件上传

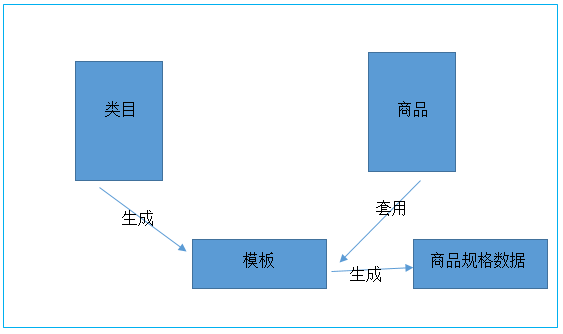
需要注意的点是对上传的文件做校验。

### 7.Fckeditor

Kindeditor上传后返回内容类型

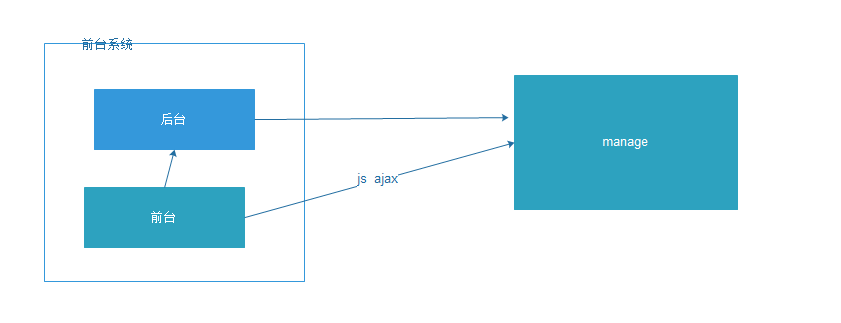


### 8.商品规格



重点是实现思路

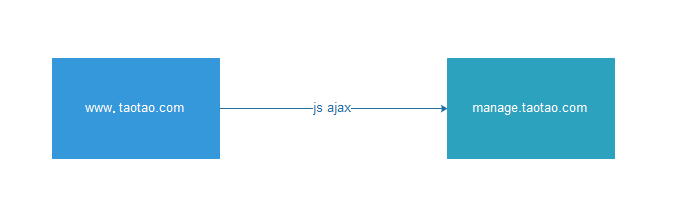
### 9.跨系统调用方式



2种调用方式：

1. Ajax，走前台js，通过jsonp来跨域，
   1. 效率
   2. 带宽
2. 后台转发请求，走后台，通过httpclient来调。
   1. 可以加逻辑(加缓存只能这条路走)
   2. 安全，接口不在公网公开

### 10.跨域



解决方案：

1. document.domain="taotao.com"
2. jsonp，部署子域名的情况

需要在SpringMVC中扩展MappingJackson2HttpMessageConverter，支持jsonp。

跨域问题，因为我们是子域名访问子系统接口的，采用jsonp解决；

### 11.CMS

内容管理系统的重点是学习CMS系统的设计思想。

### 12.Httpclient

1. 重点学习itcast-httpclient中的示例
2. 掌握spring和Httpclient的集成

### Json和java对象的互转

### 14.单点登录系统

1. 掌握单点登录系统的设计思想以及在系统架构中的位置、作用；
2. 并发压力是在通过ticket查询用户信息；解决方案？

### 15.订单ID的生成

1. 订单ID不能重复
2. 订单ID尽可能的短（占用存储空间少，实际使用方便，客服相关）
3. 订单ID要求是全数字（客服）

订单ID的生成规则：

用户ID+当前系统的时间戳

String orderId = order.getUserId() + "" + System.currentTimeMillis();

问题1：各个服务器的时间不统一。

解决：在各个服务器上做时间的统一；（运维）

问题2：在问题1解决的基础上，可能存在毫秒级的偏差情况下？

解决：修改订单生成规则：

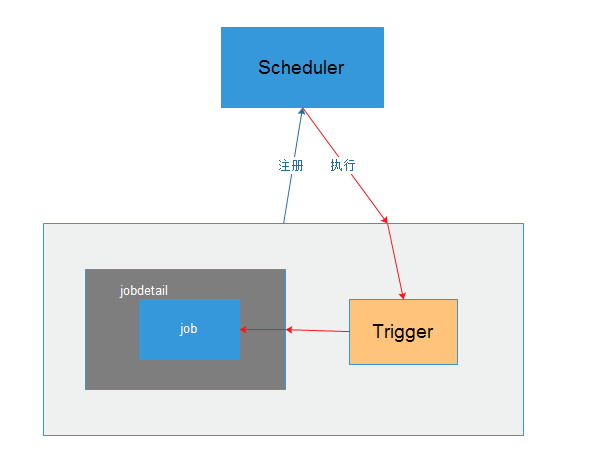
用户ID+当前系统的时间戳+随机数（3~4位） 问题：太长？ 把时间戳中的2014中的20拿掉；

目的：使用订单ID不重复；

### 16.查询订单时需要使用延迟加载

### 17.quartz

1. 基本使用
2. 了解cron触发器的规则
3. quartz和spring的集成



定时任务，job包一个jobdetail，再加上trigger，一起注册到调度器中，调度器先执行trigger， 再执行job，通过trigger触发了job，这是执行逻辑。

### 18.job和trigger关系

一个job可以有多个trigger；

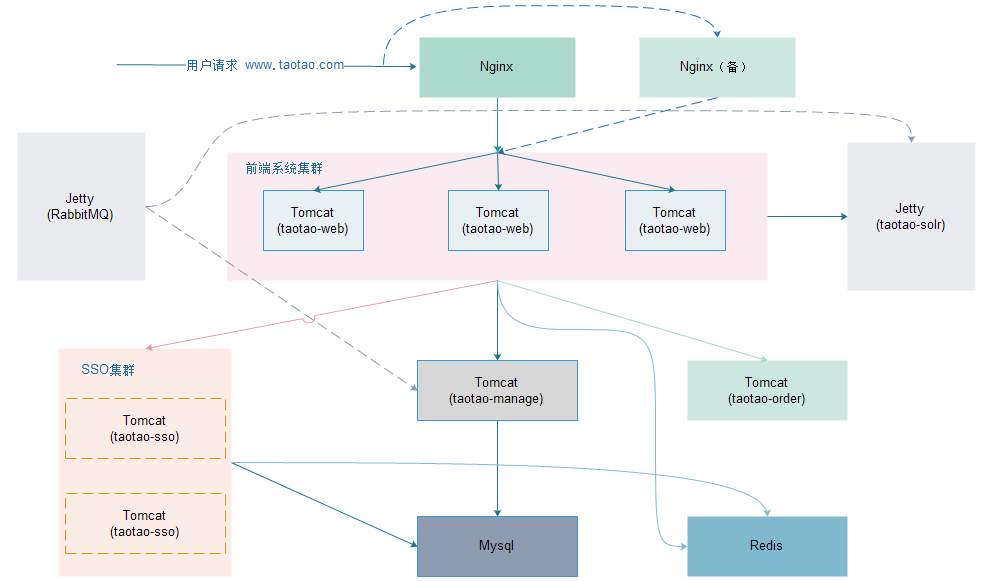
一个trigger只能有个一个job；

### 你们的架构是什么样的，能画一下吗？



### 你们的线上部署是什么样的，能画一下吗？

订单系统集群，数据库读写分离、分库分表



### 多台tomcat之间的session是怎么同步的？

不用session，我们使用单点登陆，使用redis，存在redis，生成，

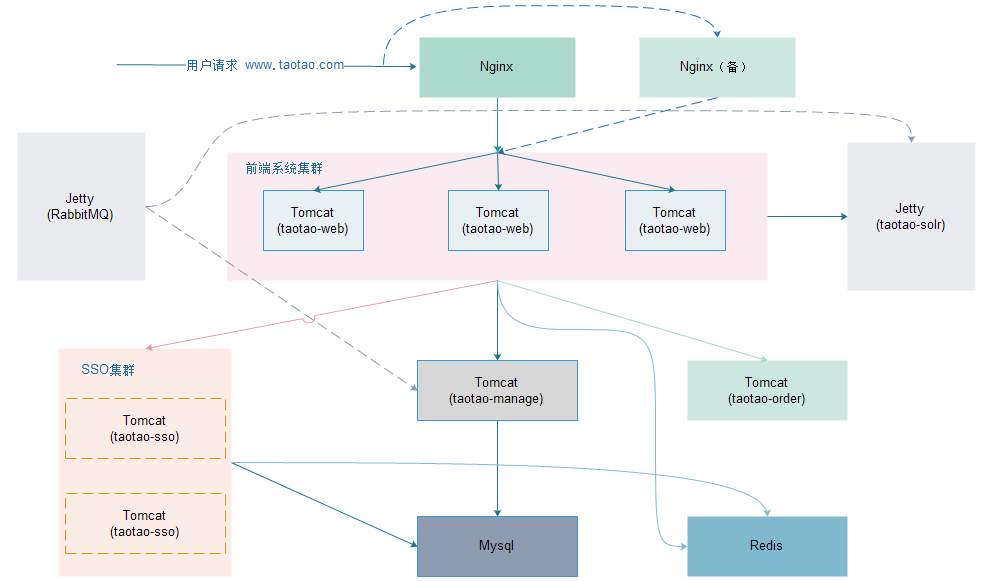
A同步到B，B同步到C。

### 如何解决大并发的问题？

集群，负载均衡，nginx(主备，一般主在工作，备闲置；资源浪费)，lvs(在2个Nginx前做一个拦截，接收后进行分工)。有问题，如果nginx挂掉，整个系统就挂了。可以主备解决，可以前面搭一个lvs。这块不是你做的，但是你知道怎么解决(非常复杂，但是必须了解。针对具体的情况去具体对待，CPU，内存，不要一刀切。)

### 你们生产环境的服务器有多少台？

面试前要数好，一般是十几到二十台。(用在哪里？这是重点)



### 数据库备份是怎么做的？有没有做读写分离 ？

主从(一主多从，主要是备份主)，每天备份，备份的文件不要放到数据库服务器上，可以FTP。要检查有效否。读写分离自己查一下，分库分表做过。

### 25.项目中的职责

在项目中主要负责相关系统的开发，主要有：

1. 后台管理系统，主要实现商品管理、商品规格参数管理、订单管理、会员管理等、CMS(内容管理系统)等，并且提供了跨域支持；
2. 前台系统，主要是面向用户访问，使用Httpclient和后台系统接口做交互，并且该系统在部署上采用集群的方式；
3. 单点登录系统，主要是提供集中用户登录凭证的集中解决方案，提供和用户信息相关的接口，比如说用户注册、查询等接口。
4. 订单系统，主要是提供和订单相关的业务接口，在订单系统了做了严格的数据校验以及高并发写的支持（这里可以说使用队列实现），

并且使用了Quartz定时任务实现对订单的定时扫描，比如说关闭超时未付款的订单；

1. 搜索系统，主要是提供商品的搜索，采用开源企业级系统Solr实现，采用了MQ机制保证了商品数据可以及时同步到solr中；
2. 会员系统，主要是维护用户的信息，已购买订单、优惠券、系统消息、修改密码、绑定手机等功能；
3. 缓存，主要是用Redis实现，并且对Redis做了集群来保证Redis服务的高可用。
4. 支付系统，主要是负责订单的支付、对账等功能，主要是对接了支付宝的接口；

### 26.你们有几台服务器

Nginx至少2台

Tomcat至少3台以上

数据库至少2台

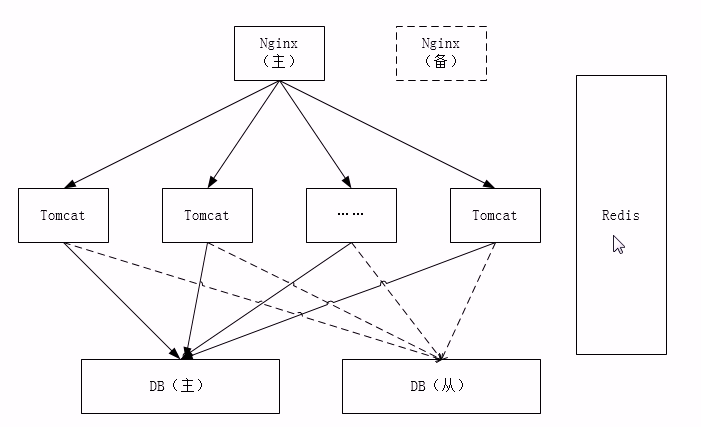
Redis至少一台

### 27.你们的服务器不止一台吧，那么你们的session是怎么同步的？

购物车用session做的话，极易问到此问题！

Session放到redis里面，使用单点登录系统。购物车设计思路：未登录(先写到cookie中，登录后写到数据库表中)；已登录(直接写到数据库，而不会写到cookie)

实际项目是不使用session的，使用redis集中处理处理数据，取代session的作用，应用在单点登录、购物车等。



### 28.你们用易宝做支付时，如果请求超时了怎么处理？

①重试，一般三次，每次重试都要停顿一会，比如，以第一次停顿1秒，第二次停顿2秒，第三次停顿3秒；

②给订单标识付款异常状态，并且发出警告(邮件、短信)给相关人员。

③写个定时任务，定时处理异常状态的订单。

### 你刚才不是说付款成功后易宝会有数据返回吗？如果付款后易宝没有返回，或者返回超时了，但是钱又已经扣了，你怎么办？

①我们请求了易宝，但是没有接受到响应，我们就认为该订单没有支付成功，并且将订单标识为异常状态；

②使用定时任务处理；

③做一个对账的任务，实时处理异常状态的订单。

### 你们怎么做退款功能的，要多长时间才能把钱退回给用户？

用户申请退款后，经过客服审核通过会将退款请求提交到易宝，具体到账时间要看易宝的处理。

### 你们WEB用的是什么容器？

应该指的是WEB服务器，应该问的是nginx或者是Apache。

### SVN主干、分支、标签开发过程流程：



### 人员配置

* 产品经理：3人，确定需求以及给出产品原型图。
* 项目经理：1人，项目管理。
* 前端团队：5人，根据产品经理给出的原型制作静态页面。
* 后端团队：20人，实现产品功能。
* 测试团队：5人，测试所有的功能。
* 运维团队：3人，项目的发布以及维护。

### 开发流程



### 系统功能



第一个问题：你前台portal采用4台服务器集群部署 那你数据库有几台服务器？那能前台高并发访问性能提上去了，那数据库会不会造成一个瓶颈，这一块你是怎么处理的？

第二个问题：你说你用了redis缓存，你redis存的是什么格式的数据，是怎么存的

第三个问题：你购物车存cookie里边 可以实现不登录就可以使用购物车 那么我现在没有登录把商品存购物车了 然后登录了 然后我换台电脑并且登录了还能不能看见我购物车的信息？如果看不到怎么做到cookie同步，就是在另外一台电脑上可以看到购物车信息

第四个问题：点一个链接访问到一个页面，这个页面上既有静态数据，又有动态数据（需要查数据库的），打开这个页面的时候就是很慢但是也能打开。怎么解决这个问题，怎么优化

第五个问题：如果用户一直添加购物车添加商品怎么办？并且他添加一次你查询一次数据库？互联网上用户那么多，这样会对数据库造成很大压力你怎么办？