目录

[介绍下dubbo和zookeeper： 5](#_Toc494351955)

[dubbo+zookeerper怎样实现session共享（在消费端）： 5](#_Toc494351956)

[为什么用dubbo+zeekeeper?介绍一下？ 5](#_Toc494351957)

[生产者和消费者是怎么交互的？ 5](#_Toc494351958)

[你看过dubbo底层吗？ 5](#_Toc494351959)

[zookeeper的作用： 6](#_Toc494351960)

[项目安全性怎么解决？ 6](#_Toc494351961)

[购物车在redis中是怎么存的？ 6](#_Toc494351962)

[问静态化的商品详情页面，如果数据库价格变动，页面会重新生成吗？ 6](#_Toc494351963)

[问购物车的商品会存到redis里面，假如有一件衬衣卖68，有一千万人将其加入购物车，而第二天老板将数据库价格改成88了，那购物车里面的价格会变化吗？ 6](#_Toc494351964)

[solr搜索引擎： 6](#_Toc494351965)

[购物车实现cookie+redis： 7](#_Toc494351966)

[redis讲解 9](#_Toc494351967)

[Redis可能出现的问题： 9](#_Toc494351968)

[Redis持久化方面： 9](#_Toc494351969)

[Redis主从同步： 10](#_Toc494351970)

[如何在项目中使用redis的： 10](#_Toc494351971)

[redis常见性能问题和解决方案： 10](#_Toc494351972)

[mongoDB主要在什么场景下使用？ 并介绍spring和momongoDB如何整合： 11](#_Toc494351973)

[Webservice + httpclient 13](#_Toc494351974)

[服务器调优 14](#_Toc494351975)

[Redis解决session共享 14](#_Toc494351976)

[nginx+tomcat负载均衡的配置(被动说) 14](#_Toc494351977)

[poi 15](#_Toc494351978)

[nginx+Tomcat 集群介绍，轮询，session共享如何实现 15](#_Toc494351979)

[数据库篇 16](#_Toc494351980)

[数据库的主从同步读写分离: 16](#_Toc494351981)

[主从同库延迟问题？ 16](#_Toc494351982)

[Sql优化: 16](#_Toc494351983)

[索引、存储过程、函数语法及各自的优缺点： 16](#_Toc494351984)

[分区和分表的区别： 19](#_Toc494351985)

[防sql注入 20](#_Toc494351986)

[数据库连接池： 20](#_Toc494351987)

[索引概述 20](#_Toc494351988)

[数据库三范式： 20](#_Toc494351989)

[Oracle存储过程基本语法 存储过程 20](#_Toc494351990)

[登录注册： 21](#_Toc494351991)

[使用aop实现mongodb日志存储，请写出实现思路和伪代码（500字） 21](#_Toc494351992)

[电商系列问题： 22](#_Toc494351993)

[产品详情页（freemarker） 22](#_Toc494351994)

[静态化介绍 22](#_Toc494351995)

[涉及到一个静态化页面ajax异步请求的问题 24](#_Toc494351996)

[如何解决跨域问题？ 24](#_Toc494351997)

[支付： 24](#_Toc494351998)

[商品管理模块 24](#_Toc494351999)

[报表统计 24](#_Toc494352000)

[统计分析模块： 24](#_Toc494352001)

[购物车及订单支付（redis）： 25](#_Toc494352002)

[每秒处理十万订单的支付系统思路 25](#_Toc494352003)

[Junit单元测试： 25](#_Toc494352004)

[框架篇： 26](#_Toc494352005)

[SpringMVC的运行原理是： 26](#_Toc494352006)

[struts2和springMVC的区别 26](#_Toc494352007)

[hibernate缓存概述: 26](#_Toc494352008)

[Hibernate和Mybatis的区别 27](#_Toc494352009)

[mybatis和ibatis的区别 27](#_Toc494352010)

[【$与#号的区别】 27](#_Toc494352011)

[Oracle和Mysql的区别 28](#_Toc494352012)

[为什么做java的web开发我们会使用struts2，springMVC和spring这样的框架? 28](#_Toc494352013)

[下面我要谈谈spring了。 29](#_Toc494352014)

[基础方面： 30](#_Toc494352015)

[BS和CS的区别以及优缺点 30](#_Toc494352016)

[五险一金，和上交比例 30](#_Toc494352017)

[HttpClient 30](#_Toc494352018)

[事务的隔离级别和传播特性 31](#_Toc494352019)

[Ajax 32](#_Toc494352020)

[Vector & ArrayList的区别 32](#_Toc494352021)

[运行时异常与一般异常有何异同？ 33](#_Toc494352022)

[HashMap的底层代码/原理 33](#_Toc494352023)

[什么是单例多例 35](#_Toc494352024)

[懒汉式单例模式： 36](#_Toc494352025)

[饿汉式单例模式 36](#_Toc494352026)

[线程安全式单例模式： 36](#_Toc494352027)

[冒泡排序： 37](#_Toc494352028)

[Sql语句： 37](#_Toc494352029)

[二叉树 37](#_Toc494352030)

[JDK版本特性！ 37](#_Toc494352031)

[接口与抽象类的区别？ 38](#_Toc494352032)

[get和post请求的区别？ 38](#_Toc494352033)

[&和&&的区别。 38](#_Toc494352034)

["=="和equals方法的区别？ 38](#_Toc494352035)

[Integer与int的区别 39](#_Toc494352036)

[面向对象的特征有哪些方面 39](#_Toc494352037)

[linux中的命令： 39](#_Toc494352038)

[Jvm相关 39](#_Toc494352039)

[Jvm优化： 40](#_Toc494352040)

[多线程介绍、列举线程池和业务场景： 41](#_Toc494352041)

[jQuery的10个常用方法,常用选择器(五大类),ajax常用的属性 42](#_Toc494352042)

[Tomcat优化 42](#_Toc494352043)

[Hibernate优化： 42](#_Toc494352044)

[序列化和反序列化 43](#_Toc494352045)

[集群 43](#_Toc494352046)

[非专业知识： 44](#_Toc494352047)

[大学情况： 44](#_Toc494352048)

[上家公司： 45](#_Toc494352049)

[有没有上过保险： 45](#_Toc494352050)

[没有上五险一金的原因： 45](#_Toc494352051)

[五险一金的名称： 45](#_Toc494352052)

[三个词形容自己： 45](#_Toc494352053)

[谈谈你的对软件行业发展前景的理解： 45](#_Toc494352054)

[为什么离职： 45](#_Toc494352055)

[如何看待你上家公司： 45](#_Toc494352056)

[你凭什么要这么高的薪资? 45](#_Toc494352057)

[四年涨薪： 46](#_Toc494352058)

[跳槽的看法: 46](#_Toc494352059)

[你的五年规划 46](#_Toc494352060)

[谈谈你对加班的看法： 46](#_Toc494352061)

[分库分表： 46](#_Toc494352062)

[最后你还想有什么了解吗？ 46](#_Toc494352063)

X经理，你好！我叫，今天来咱公司面试JAVA开发工程师，之前在北京路都科技有限公司任职，从事这一行已经有4个年头了。这几年开发，主要涉及的行业项目包括拍拍贷网贷系统，深圳网上天虹商城系统，码头帮物流管理系统，中国电信人力资源管理平台等。在开发过程中，也用过好些框架，比如：dubbo+zookeeper、springboot、springmvc、spring、Mybatis等框架，熟练掌握框架之间的整合技术。

有时候因为项目需求或是为了开发的高效性，自己也会研究一些技术，使用一些常用的主流java技术，例如：solr全文检索，redis缓存框架、jms消息队列，activieMQ异步调用，shiro安全认证框架，activiti工作流等。此外还有一些比较简单的POI导入导出、highcharts报表，restful、HttpClient远程调用，powerdesigner建模，spring定时器。前端的技术也研究过一些。如jquery、Ajax、Easyui、BootStrap。

在秒创后期担任的是高级java工程师，主要的任务是：与项目经理到客户现场沟通并确认需求，并参与需求文档的编写。在设计阶段，参与详细设计，包括框架的一些基类、工具类的封装，使用powerdesigner完成数据库的设计。对数据库设计有一定的经验，常用的数据库包括mysql、oracle等关系型数据库，还有mongodb、redis等非关系型数据库，并在linux上搭建集群和哨兵和部署项目。编码过程中，主要负责一些功能模块的编码实现，并使用junit做单元测试。项目管理上，与项目经理沟通协调过项目的进度，使用maven对项目进行管理，还帮助其他开发人员解决一些开发过程中的问题。

平时呢还喜欢上一些技术网站比如csdn、开源中国、博客园等这些网站来提高自己的知识量和扩展知识面，毕竟时代在发展，科技在进步。要跟上时代的步伐，避免被淘汰啊！

（在闲暇时间，我也会去访问一些像CSDN 、开源中国、博客园等这些技术网站，研究一些新的技术点，来增加自己的行业知识。在研究这些技术的同时呢也总结出了一点点经验比如我研究时会根据内容在云笔记上进行分类，比如将一些关于新技术方面的技术博客的链接放在一个文件夹，然后将用到的一些jar包啊js之类的放到一个文件夹，然后通过看这些资料自己写一个小的demo例子放在一个文件夹下，最后再把自己研究的过程以及心得，还有容易忽略的问题写在一个总结文档的文件夹下， 然后与项目组的同事一起分享交流、共同进步，我一直坚信，一个人的能力是有限的但是团队的能力是无限的，只有这样才能来给公司创造更大的价值）

**（1）项目名称：优配良品网上商城**

* 开发时间：2016/12-至今
* 项目描述：

项目框架:

后台： shiro + Spring + Mybatis + SpringMVC(基于注解) + Bootstrap + OScache + Solr + ActiviMQ(消息队列) + dubbo + zookeeper + Echarts + WebService

前台： Spring + Mybatis + SpringMVC(基于注解) + FreeMarker + Redis + Solr集群 + spring定时器 + jQquery

该项目主要有会员注册及登录、商品分类展示、商品信息检索、购物车、生成订单、订单查询、商品排行、反馈留言、商品类别管理、商品品牌展示、商品管理、会员管理、订单管理、新闻管理等。包括严格的权限管理部分，客服登录系统后可以对客户所下订单进行查询和取消备注等操作，在取消的同时会对客户所下订单进行积分，款项返还等操作。

* 开发环境：idea + tomcat7.0 + jdk1.7 + Maven + git
* 使用技术：SpringMVC4.0 + Spring3 + Mybatis + dubbo + zookeeper + ActiviMQ + Solr + Nginx + FreeMarker + ECharts
* 运行环境：linux + Tomcat + Mysql + Redis + MongoDB + JDK1.7
* 本人角色：高级程序员
* 责任描述：在项目中，我参与架构选型与搭建框架，技术攻关，并且负责后台权限管理、商品管理、报表统计（ECharts）、商品信息检索引擎搭建（Solr）及部分核心代码的编写（大数据量poi导出、shiro安全认证整合）。负责前台的产品详情页（FreeMarker，Redis）、购物车及订单支付（ActiviMQ + WebService）、产品评论（MongoDB）、支付成功的短信提醒、服务器搭建（Nginx负载均衡Tomcat7.0集群）等，后期还对项目进行优化和安全测试，在项目中担任高级工程师，并带了一个新人。

项目总体采用的是Dubbo+zookeeper的分布式框架，可以理解问拆分成了三个项目，前端项目、服务端项目和缓存项目，前端项目包含页面、静态资源和控制层；服务端项目包含业务层和数据库操作层；缓存项目缓存前端项目和服务端项目公用的数据。用户请求进入前先通过负载均衡设备进行请求分发，采用nginx的负载均衡实现动静分离，首先呢我们使用nginx+keepalived双机热备的方式来实现负载均衡调度器的高可用性，其中keepalived主要是解决了nginx的单点故障，用来检测服务器状态，如果一台web服务器挂了，keepalived将检测到，并将有故障的机器剔除掉，然后人工进行修复。通过nginx中server标记下添加location规则信息，将css，js这些动态资源交给nginx来处理，将动态请求通过proxy\_pass

## 介绍下dubbo和zookeeper：

Duboo是一个分布式框架，zookeeper是duboo生产者暴露服务的注册中心。起一个调度和协调功能，当然注册中心也可以用redis 或者duboo自带的Multicast 来注册。

Duboo 通信方式采用长链接方式，所以当spring启动后链接就接通，duboo的消费者和生产者就可以直接调用。性能上高于其他http协议的请求。（httpclient数据属于短请求，一次请求，一次响应，dubbo通道一旦建立（一旦连接），一直处于联通状态）dubbo基于tcp/ip协议的，交互性能非常高，支持匿名传参，隐式传参，泛化调用（隐式传参：不是以括号的形势传参，将参数存储到当前请求里面，到了服务端再从请求里面拿出来，request不能传参。泛化调用：是指采用一个service接口和一个service实现类来实现多个方法的调用）当时是为了解决单个服务器站点的压力，将项目拆分成页面加controller属于消费者，service+dao属于生产者，所有生产者暴露的端口都注册在zookeeper里面。这时候，消费者要调用生产者去zookeeper中取就可以了。所以我们部署了多套生产者，所有的消费者的请求可以由多个生产者去提供，具体由哪个生产者提供可以由zookeeper的配置去决定。如果某个生产者挂掉，zookeeper会加压力导向其他生产者，当这个生产者恢复状态的时候。Zookeeper会重新启用它。因为我们用n的执行,如果想让单个tomcat执行的action---service—dao 请求又多个tomcat来执行就可以使用 zookeeper+duboo 这时候一般是一个tomcat里面部署的是jsp+action所有的service接口都注册到了zookeeper里面，action去zookeeper里面通过duboo去调用哪个 service+dao 的组合。而且service+dao的组合可以配置多套。一套挂了其他的service+dao组合可以继续使用

dubbo框架的体系结构有5个核心组成部分，分别是提供者provider,它的作用是为消费者提供数据。注册中心registry,它的作用是用来注册和发现服务。消费者consumer,它的作用是调用远程提供者提供的服务。监控中心Monitor用来统计服务的调用次数以及调用时间，还有container用来充当容器来加载，运行服务提供者。新建dubbo-provider.xml配置文件，通过dubbo:application配置提供者应用名，通过dubbo:registry配置注册中心的地址，通过dubbo:protocol配置协议，以及通过dubbo:service来暴露要发布的接口。

最后我们在需要使用dubbo接口的项目中配置消费者信息，新建dubbo-consumer.xml文件，通过dubbo:application配置消费者应用名,通过dubbo:registry指明要订阅的注册中心地址，通过dubbo:reference指定要订阅的服务接口。除此之外考虑到dubbo的健壮性和性能我们对它的参数项进行的调优。通过在dubbo:protocol中threadpool="fixed" threads="200"来启用线程池，通过在dubbo:service中connections=5来指定建立长连接的数量。

配置dubbo集群来提高健壮性以及可用性。dubbo默认的集群容错机制是Failover即失败自动切换，默认的重试次数为2，可以通过retries调整。dubbo默认的负载均衡策略是Random随机，可以按权重设置随机概率。我们在写完dubbo提供者之后，为了测试接口的正确性，我们会进行直连测试。首先会在提供者端，通过将dubbo:registry的register设置为false,使其只订阅服务而不注册现在正在开发的服务；

在消费者端，通过设置dubbo:reference的url，直连提供者进行测试。被动说：

所谓dubbo集群就是将dubbo的提供者部署多份，在不同的机器上或者说在同一台机器上用不同的端口号。从而在启动时可以向注册中心进行注册，这样结合dubbo的集群容错策略以及负载均衡策略可以提高可用性。

dubbo负载均衡策略：随机,轮询，最少活跃调用数。

dubbo的集群容错:失败自动切换,快速失败,失败安全。

## dubbo+zookeerper怎样实现session共享（在消费端）：

我们的消费者只有一个模块，所有的请求首先都是进入这个模块里面，生产者有多个模块，session都在消费者这个action里面，所有的请求都是首先进入这个项目里面，我们的生产者有多个模块。如果有多个消费者的情况下，会存在session共享问题，我们可以将session的id作为key值，用户对象作为value值存储到redis里面。当每次发送的请求的时候，拿着浏览器的session的id去redis里面取，如果能取到，证明用户已经存在，如果不能取到，就重新登录。

## 为什么用dubbo+zeekeeper?介绍一下？

实现了分布式部署，如果不用的话，传统的项目就是action>servcie>dao,有三部分的请求，单个儿tomcat，用了dubbo+zookeerper后，分成了消费者和生产者，划分成不同的模块，有几个模块就有几个tomcat，大大降低了tomcat的压力，而且后期随着访问量的增加，我们可以不断的增加生产者，每个单个web服务节点者所受到的压力明显降低。

## 生产者和消费者是怎么交互的？

dubbo首先向zookeerper暴露端口，消费者向它们这边订阅服务通过zookeerper，消费者发送请求，zookeerper用来调度，调度的方式有三种，轮循，随机，把这些请求都拿给生产者，如果其中一个生产者挂掉，如果有新的生产者，会把请求分发给新的生产者。

## 你看过dubbo底层吗？

生产者跟消费者之间相当于长连接长请求，传统的request请求，response响应，请求一次，响应一次，开关一次，dubbo是基于 tcp/ip协议的，其他的源码闲的时候，基本上通过maven来看过没有刻意的去了解，因为平时这个业务开发状况很繁琐。

## zookeeper的作用：

dubbo是把项目实现分离，分为消费端跟服务端，在消费者这边的配置文件中添上zookeeper地址，让它连上zookeeper，每个服务端都暴露一个端口给zookeeper，这样就实现服务端在zookeeper中注册，消费者发送请求给zookeeper，zookeeper接收请求，zookeeper再分发给这些服务端。

## 项目安全性怎么解决？

是指攻击安全还是访问量过大安全还是数据安全？

如果是攻击性的话，input框都采用了js验证，用java代替重复效验，避免输入不必要的空格或者sql注入。

如果是访问的安全的话，页面做了大量的页面静态化，缓存，通常访问量不会增加服务器造成负担。

如果是数据安全这一块的话，系统用的是linux，数据库的账号密码都做了加密处理。

如果是数据遗失或是自然灾害，数据库做了读写分离，而且一个主库，两个从库，其中一个从库做了备份，而且是定时备份。

在登录过程中，为了防止程序直接登录，也做了一些验证码，包括支付的资金账户安全，我们的系统没有钱，主要是通过超链接将钱存储到网络支付宝qq账号，最主要还是支付宝的安全和网民的安全,而且这个账号只能出不能进。

## 购物车在redis中是怎么存的？

我们的购物车，用的是redis 来实现的。 当加入购物车的时候 用户id作为redis 的key，产品集合作为redis的value。商品存的是 ，商品id 商品名称，和商品购买数量。

当加入商品到购物车的时候，首先判断当前用户id对应的的产品集合里面是否含有当前产品，有则数量加一。没有则新添加该商品。

## 问静态化的商品详情页面，如果数据库价格变动，页面会重新生成吗？

答：不会重新生成。我们的价格和数量是来自redis缓存的。一般我们将商品更新到，redis中，key是表名加id例如 sm\_product\_proid , value是产品对象。存储价格数量。

## 问购物车的商品会存到redis里面，假如有一件衬衣卖68，有一千万人将其加入购物车，而第二天老板将数据库价格改成88了，那购物车里面的价格会变化吗？

答：会变化。我的购物车用户id做为key，value是产品集合，而且默认过期时间是7天，产品集合只存储产品id ，数量，名称。 当显示该商品的时候我需要用产品id 去其他redis 库里面获取价格。其他的redis库里面已经存储了所有产品id作为key，value是商品 价格数量。当老板变动数据库价格的时候我只需要更新，单独存储的这边redis产品。

## solr搜索引擎：

在整个项目中实现商品搜索功能电商项目中，因为用户有时候不是多么清楚他所需要的东西的名称或者商店的名称，有时候仅仅是只知道他所需要的商品是干嘛用的，又或者是有多个要求，为了满足用户的诸多需求更准确的查询到用户所需要的商品，所以就用到了Solr搜索引擎

solr是一个基于Lucene的全文搜索服务器，相比Lucene更高效、使用更便捷，在进行模糊匹配的时候，他可以 用来替代数据库中的like ,从而在匹配准确性以及性能进行大幅度的提高。因为官方的分词器对中文支持不好所以使用了第三方的IK分词器来进行分词。在建立索引的时候我们通过在schema.xml配置IK分词器来完成中文分词。从而实现了高亮显示关键词，分页，排序，多字段，多条件的高性能搜索。

其实目前有很多优秀的中文分词组件 ：像mmseg4j，IK Analyzer ，Paoding

通过查询资料发现用 IKAnalyzer比较好IK 比其他中文分词维护的勤快，和 Solr 集成也相对容易。

其中我们在配IK分词器的时候，当时遇到了三个问题：

分词器版本与solr服务版本的匹配：

当时用的solr是4.10.4版本的因为这个版本比较好用，稳定，5.以上的版本bug比较多，配IK分词器的2012FF\_u1版本，开始时用的是u6,然后配完效果出不来，上网查了好多资料，又问了之前加的一个搜索引擎群，才知道是版本问题，u3,u5,u6的都不好用，只有u1的和solr4.10版本的合适，改完之后问题确实解决了

加入分词器的schema文件要把version改为1.5基本的CRUD操作都可以。

但是搜索却只能全字匹配或者只能单个字符匹配出结果。这是绝对不能容忍的。定位问题接近了一天，找有经验的同事给 排查也没排查出来问题。最后我自己一点一点比对multicore文件夹下的配置文件跟F:\solr\solr-4.6.0\example\solr \collection1这个文件夹下的配置文件的配置区别。当我把schema.xml的version属性从1.1升到跟collection1下的相同文件的1.5一致之后。重启服务器，问题解决了！、

分词器的词典编码格式为UTF-8无BOM格式

在从数据中取数据生成索引的时候，因为表中的数据量比较大，防止一次取出所导致内存溢出问题，我采用了分段批量提取的方式进行，

此外我们为了提高solr搜索的性能对其进行了主从配置。

1. 我们solr使用的是solr4.7版本

2. 通过修改schema.xml来添加要进行索引的字段以及增加ik分词器

3. 通过solrj将数据库中的数据生成solr中的索引文件，注：solrj是java程序调用solr服务所用的jar包。

4. 通过在solrconfig.xml中配置requestHandler name=“/Replication”来进行主从同步的配置，在从solr中通过masterUrl指明要从哪些主solr服务器中同步数据

我们为了提高用户体验度还使用了solr的spellCheck可以用于查询输入的自动完成功能auto-complete。他是基于动态代码方式建立内容，suggestion可通过读文件方式建立内容，并有点击率排序。使用facet在查询结果上根据分类添加了count信息, 然后用户根据count信息做进一步的查询, facet的主要好处就是可以任意对搜索条件进行组合, 避免无效搜索, 改善搜索体验.

## 购物车实现cookie+redis：

一、未登录状态下添加商品到购物车

在不登陆的情况下也可以添加购物车。把购物车信息写入cookie。

优点：

1、不占用服务端存储空间

2、用户体验好。

3、代码实现简单。

缺点：

1、cookie中保存的容量有限。最大4k

2、把购物车信息保存在cookie中，更换设备购物车信息不能同步

实现思路：

（1） 从cookie中获取商品列表信息（单独提出来写成个通用的方法）

（2） 遍历购物车列表，判断需要添加的商品在购物车列表是否存在

（3） 商品存在的话，那么取出该商品原来的数量+添加的数量作为该商品现在的数量

（4） 如果商品不存在，那么调用服务，根据传来的商品id查询商品数量，设置商品的数量为页面传来的数量，取商品的第一张图片(购物车列表只展示一张图片)。

（5） 把修改后的购物车列表重新存入到cookie中

（6） 返回逻辑视图”cartSuccess”

2、展示购物车列表

单击“去购物车结算按钮”向服务端发送请求，服务端应该返回逻辑视图”cart”

请求地址:8090/cart/cart.html

返回逻辑视图:”cart”也就是购物车列表页面

3、登录状态下购物车列表页面修改商品数量

购物车列表页面单击”+”,”-”会向服务端发送ajax请求。

页面需要根据调整的数量重新显示商品总计（已经实现了也就是输入框的值\*价格）和小计（用js，待实现）

服务端要求修改cookie中对应商品的数量

4、未登录状态下删除购物车商品

（1）从cookie中获取购物车列表   
 （2）遍历，查找到要删除的商品   
 （3）将该商品从购物车列表移除   
 （4）更新后的购物车列表重新写入cookie   
 （5）重定向到购物车列表页面

**二、登录状态下购物车功能的实现**

功能分析：   
 1、购物车数据保存的位置：   
 未登录状态下，把购物车数据保存到cookie中。   
 登录状态下，需要把购物车数据保存到服务端。需要永久保存，可以保存到数据 库中。可以把购物车数据保存到redis中。   
 2、redis使用的数据类型   
 a) 使用hash数据类型   
 b) Hash的key应该是用户id。Hash中的field是商品id，value可以是把商品信息转换成json   
 3、添加购物车   
 登录状态下直接把商品数据保存到redis中。   
 未登录状态保存到cookie中。   
 4、如何判断是否登录？   
 a) 从cookie中取token   
 b) 取不到未登录   
 c) 取到token，到redis中查询token是否过期。   
 d) 如果过期，未登录状态   
 e) 没过期登录状态。

实现：

1、登录拦截器   
 几乎在购物车所有功能执行 都要判断用户是否登录。利用aop思想，应该编写个拦截器，来判断用户是否登录。登录的话用户信息需要存在request域中   
 （1） 从cookie中取token   
 （2） 判断token是否存在   
 （3） 不存在，说明用于未登录，放行   
 （4） 如果token存在，调用服务，根据token从redis中取用户信息   
 （5） 取不到用户信息，说明已经过期，放行   
 （6） 取到了用户信息，说明用户已经登录，用户信息存到request中   
 （7） 放行

2、登录状态下，商品添加功能实现

向redis中添加购物车

比未登录多一个登录判断而已

3、登录状态下，商品列表展示   
分析：   
（1）从cookie中取购物车列表   
（2）判断用户是否登录   
（3）用户已经登录的话，则调用服务层，合并cookie中的列表和redis中的列表。存入到redis中。   
（4）同时删除cookie中的购物车列表   
（5）根据用户id，调用服务查询redis中所有的商品，返回购物车列表。   
（6）未登录状态还是跟前面一样   
（7）将列表绑定到参数，返回购物车列表页面

4、登录状态下修改购物车商品数量   
分析   
单击”+”,”-”修改商品的数量的时候,要求redis中该商品的数量发生改变   
（1） 根据用户id，商品id从redis中取出对应的商品   
（2） 设置商品的数量   
（3） 该商品更新到redis中

5、登录状态下，删除购物车商品   
分析   
单击删除的时候，删除redis中该商品。重定向到列表页面   
（1） 直接用jedisClient的del的方法根据用户id跟商品id 商品   
（2） 返回成功

## redis讲解

另外为了保证高并发访问量大的情况，避免过多的连接一下子查询到数据库造成数据库的崩溃呢，我们采用了redis来实现数据的缓存，这样呢就会查询数据的时候，先从redis缓存中查询数据是否存在，如果缓存中存在我们就直接从缓存中取，这样可以减少多数据库的访问，如果缓存中不存在数据再去数据库查询，并且将查询出来的数据添加到缓存中，因为redis的查询速度是相当快的（11000次/s），另外为了保证redis服务器的安全通常会在redis.conf中绑定具体的ip地址这样只有该地址才能访问redis服务器，并且设置密码，为了保证redis不会因为占用内存过大而导致系统宕机，通常在将redis当做缓存服务器使用时，设置存储数据的过期时间，并且通过设置maxmemory【最大内存】和maxmemory-policy【数据清除策略】为allkeys-lru来达到预期的效果。我在项目中通常使用redis来充当缓存服务器来缓存分类列表，品牌列表，热销商品，推荐商品以及该商品的关联商品等等。使用了jedis作为客户端，并考虑到性能问题使用了jedis连接池。考虑到redis服务器的高可用性，我们做了redis的主从复制，并且通过加入哨兵来使redis主服务器宕机时，从服务器自动转换为主服务器继续提供服务。

## Redis可能出现的问题：

1：缓存穿透的问题：

一般出现这样的问题，是因为当我们查询一条肯定不存在的数据的时候，缓存中没有，就会透过缓存来查询数据库，数据库也不存在，这样就不会将值保存在缓存中，最后还是缓存和数据库中都没有，如果一直访问这条数据。我们就对数据库进行频繁的查询，给数据库带来压力；

解决方法：当查询的时候，如果缓存和数据库中都没有，我们就将这个数据以空的形式存放在缓存中，（或者是给一个false的标示）这样就不用去数据库就可以知道不存在，减少对数据库查询的次数，当我们这个值发生改变的时候，我们在重新进行赋值；

2：并发情况：当我们大量访问都是查询一个缓存中没有的数据时，这样就会都去数据库中进行查询，可能会造成数据库的宕机；

解决方法：在查询的时候，我给他添加了一个同步锁，只有第一条数据去数据库中查并且返回到redis中后才能查询，这是数据库中已近存在了值，这样也可以避免；

3：雪崩：大量数据的访问时间失效，这样用户就会访问到数据库，第一台数据库崩溃了，访问就会到第二台数据库进行查询，这样会导致第二台的也崩溃；

解决方法，就是设置失效时间时，不要一起失效，或者是设置在访问量少的时候，或者设置为永远不失效；

## Redis持久化方面：

redis的一大特点就是可以将数据进行持久化，在一定程度上确保了数据的安全性，但不是绝对的；

首先持久化分为rdb（快照持久化）和aof（精细持久化）；

快照持久化，是默认开启的；会自动保存数据，当启动时会在文件夹中生成dump.rdb文件；存放持久化后的数据；

当然我们也可以设置持久化的频率，在redis.conf文件中通过save进行设置，默认有三种情况，每秒超过一万数据或每5分钟有10条数据的时候再或者每15分钟有1条记录，都会执行快照持久化，

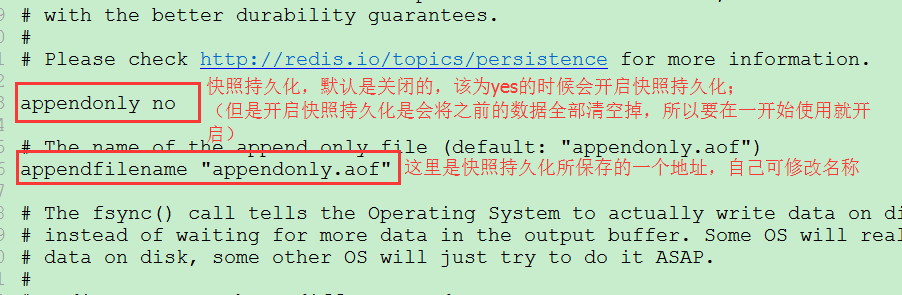
当然也可以通过bgsave的方法来手动进行一个快照持久化；（也可以通过ip和端口号就给别人进行手动持久化）；

如果频繁的快照持久化，会降低性能和效率，

但是这样也出现了一个问题，就是当一分钟内如果有一万条数据时，不会提交，但是在下一次提交之前，停电了，这样就会丢失掉这些数据；

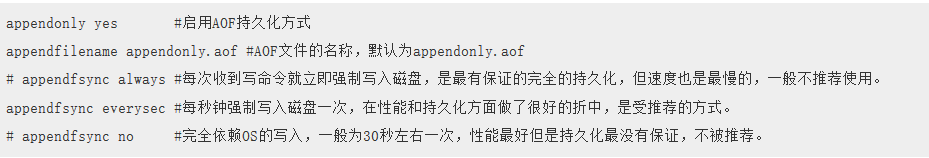
当时想到的解决方法呢就是和（AOF）精细持久化进行一个结合，达到一个秒级的持久化；

这个持久化需要我们手动进行开启，（注意，AOF开启之后，之前的数据都会丢失，所以要在一开始使用时就要配置好）开启的方法就是在配置redis.conf，将appendonly 改为yes；同时还可以更改文件名称；然后重新启动服务，这时精细化持久化就启动好了；



在快照持久化的同时，我们进行精细持久化，

比如，我们每隔一个小时进行一次快照持久化，这中间我们添加精细持久化；当55分的时候宕机了，我们还是可以通过RDB和AOF来恢复我们的数据，尽可能减少了损失；等到一个小时以后我们进行了快照持久化，数据就会保存在rdb的文件中，我们就可以将aof的持久化文件重新开始；



关于精细持久化也存在三个配置，

always只要发生改变就会发生持久化，这样是很浪费性能的，不推荐使用（我们的CPU不用干其他工作了，就在这里看着持久化）

everysec每秒进行一次快照持久化；推荐使用这种

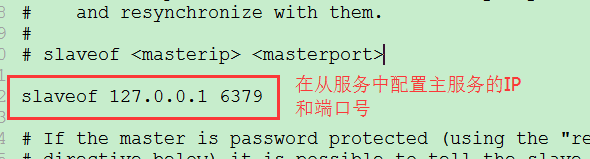
no 是不确定，看服务器的使用情况（心情）来定（不安全，不推荐）

我们还可以通过把rdb文件和aof文件复制到另外一个redis服务器上，就可以共享数据；

## Redis主从同步：

我们还可以对redis服务器配置主从同步，来减轻redis服务器的压力，

修改从服务对应的主服务地址（修改时可以直接搜索slaveof关键字）



这样访问时会读写分离，查询的时候会走从服务器，而更新操作的时候走主服务器；

与memcached的区别 （问到再说）

1，redis并不是把所有的数据都存放在内存，还可以将一些长期不使用的值存放在磁盘当中。而memcached只能存放在内存当中的；

2，redis的数据类型比memcached的数据类型丰富，除了string外，还有set,list,sortd set，hash等；

3：redis是可以持久化数据的，在一定程度上避免了数据的丢失，而memcached的安全性较低，一旦遇到停电，宕机的情况数据就会丢失；

4设置过期时间，memchched是在set key的时候直接设定时间（memcache.set(key,value,new Date(50)) 设置为50秒过期），而redis设置过期时间是通过expire来设置的；

## 如何在项目中使用redis的：

我们的项目采用的是maven来进行管理的，可以通过maven来进行包的依赖管理，保证jar包版本的一致性，所以只需要在pom.xml里添加对应的<dependency>信息即可自动下载jar包，

1：在项目中我们是通过jedis来操作redis的，jedis是java提供的用来操作redis的API

2：我们需要导入一些相关的jar包，目的是为了一些工具类的使用；

3：配置redis.xml文件,

1）我们配置了连接池；设置了最大链接数，最大空闲数；然后用静态语句块初始化连接池信息，在获得jedis实例

2）配置Jedis ConnectionFactory，通过IP和端口号来链接相对应的redis数据库；

3）配置了模板，并且在模板中引入了redistemplate的相关jar包；

4：在web.xml文件中引入了 redis.xml文件；

5：我们封装了相对应的工具类；通过操作模板来调用其中的方法，进行增删改查；在service层中获得jedis的实例，通过jedis实例来调用Redis中set类型对应的方法，就可以实现在Redis缓存中放入和取出查询到的商品的信息了，需要注意的是redis里只能存储字节，所以相对应的实体类要实现serializable这个接口；

## redis常见性能问题和解决方案：

(1) Master最好不要做任何持久化工作，如RDB内存快照和AOF日志文件

(2) 如果数据比较重要，某个Slave开启AOF备份数据，策略设置为每秒同步一次

(3) 为了主从复制的速度和连接的稳定性，Master和Slave最好在同一个局域网内

(4) 尽量避免在压力很大的主库上增加从库

(5) 主从复制不要用图状结构，用单向链表结构更为稳定，即：Master <- Slave1 <- Slave2 <- Slave3...

这样的结构方便解决单点故障问题，实现Slave对Master的替换。如果Master挂了，可以立刻启用Slave1做Master，其他不变。

还有就是Redis的秒杀场景

使用的是Redis的事务功能，先通过watch（）监控一个字段，且这个字段为0，之后对于秒杀数进行判断。当监控字段小于秒杀数时在IF条件内部内开启事务通过multi,当有人抢购成功的时候这个监控字段加1，且通过Exec执行事务，事务执行成功则抢购成功，否则抢购失败。抢购成功之后将相应的用户也存储到数据库中然后进行相应的订单处理。当监控字段大于秒杀数的时候也就是抢购完毕。无法进行抢购。

缓存的策略是每次数据在增删改之后就更新，主要是为了避免修改之后缓存更新不及时造成脏数据的出现

## mongoDB主要在什么场景下使用？ 并介绍spring和momongoDB如何整合：

mongodb是非关系型数据库，他的存储数据可以超过上亿条（老版本的mongodb有丢数据的情况，新版本不会有，网上说的），mongodb适合存储 一些量大表关系较简单的数据

，例如用户信息，用户注册信息，公司注册信息，留言，评论，操作日志，mongodb还能用分布式文件存储信息，我们主要用mongodb来存储我们项目里面的操作日志（银行的付款转账记录，角色权限的变动日志），我们主要是结合aop来使用的，首先我们来配置一个aop的切面类，再给aop的使用规则，哪个类里面的哪个方法使用当前切面类，利用后置通知类获取当前方法的操作日志，将操作日志存储到mongodb，然后进行分类管理查看。利用后置通知传到数据库。

从目前阿里云 MongoDB 云数据库上的用户看，MongoDB的应用已经渗透到各个领域，比如游戏、物流、电商、内容管理、社交、物联网、视频直播等，之所以采用mongodb是因为:

MongoDB是NoSQL的非关系型数据库，他的存储数据可以超过上亿条（老版本的mongodb有丢数据的情况，新版本不会有，网上说的），mongodb适合存储 一些量大表关系较简单的数据,易于扩展，可以进行分布式文件存储，适用于大数据量、高并发、弱事务的互联网应用，

以下是几个实际的应用案例:

游戏场景，使用MongoDB存储游戏用户信息，用户的装备、积分等直接以内嵌文档的形式存储，方便查询、更新

物流场景，使用MongoDB存储订单信息，订单状态在运送过程中会不断更新，以MongoDB内嵌数组的形式来存储，一次查询就能将订单所有的变更读取出来。

社交场景，使用MongoDB存储存储用户信息，以及用户发表的朋友圈信息，通过地理位置索引实现附近的人、地点等功能

物联网场景，使用MongoDB存储所有接入的智能设备信息，以及设备汇报的日志信息，并对这些信息进行多维度的分析

视频直播，使用MongoDB存储用户信息、礼物信息等

在我们实际项目中：

1. 我们主要用mongodb来存储我们项目里面的操作日志（银行的付款转账记录，角色权限的变动日志），我们主要是结合aop来使用的，首先我们来配置一个aop的切面类，再给aop的使用规则，哪个类里面的哪个方法使用当前切面类，利用后置通知类获取当前方法的操作日志，将操作日志存储到mongodb，然后进行分类管理查看。利用后置通知传到数据库。

2. 我们在项目中使用它来存储电商产品详情页的评论信息（评论id,商品id,标题，评分，内容，评论人信息，评论的发布时间，评论标签组）并且为了提高可用性和高并发用了3台服务器做了mongodb的副本集，其中一台作为主节点，另外两台作为副本节点，这样在任何一台mongodb服务器宕机时就会自动进行故障转移，不会影响应用程序对mongodb的操作，为了减轻主节点的读写压力过大的问题，我还对mongodb副本集做了读写分离，使写操作在主节点进行，读取操作在副本节点进行。为了控制 留言，我们留言的界面设置在了订单状态，只有状态为5，也就是交易成功收货后才能评论，并在评论成功后将订单状态改为6

【补充：如果问到副本集是怎么搭建的，就说我们有专门的运维人员来负责搭建，我只负责用Java程序去进行操作】

MongoDB 副本集（Replica Set）是有自动故障恢复功能的主从集群，有一个Primary节点和一个或多个Secondary节点组成。

在副本集的环境中，要是所有的Secondary都宕机了，只剩下Primary。最后Primary会变成Secondary，不能提供服务。

副本集的同步和主从同步一样，都是异步同步的过程，不同的是副本集有个自动故障转移的功能。其原理是：slave端从primary端获取日志，然后在自己身上完全顺序的执行日志所记录的各种操作（该日志是不记录查询操作的）

Spring和mongodb整合步骤：

1. 添加依赖的jar包

要想整合Spring和Mongodb必须下载相应的jar，这里主要是用到两种jar一种是spring-data-document和spring-data-commons两种类型的jar

2.定义实体类

3.Spring配置

新建application.xml配置文件

这个必须要的引入mongodb标签xmlns:mongo="http://www.springframework.org/schema/data/mongo"

在配置文件中加入链接mongodb客服端

<mongo:mongo host="localhost" port="27017">

</mongo:mongo>

注入mogondb的bean对象

<bean id="mongoTemplate" class="org.springframework.data.document.mongodb.MongoTemp">

<constructor-arg ref="mongo"/>

<constructor-arg name="databaseName" value="db"/>

<constructor-arg name="defaultCollectionName" value="person" />

</bean>

<bean id="personRepository" class="com.mongo.repository.PersonRepository">

<property name="mongoTemplate" ref="mongoTemplate"></property>

</bean>

4.然后编写操作mongodb增删查改的接口

public interface AbstractRepository {

public void insert(Person person);

public Person findOne(String id);

public List<Person> findAll();

public List<Person> findByRegex(String regex);

public void removeOne(String id);

public void removeAll();

public void findAndModify(String id);

}

再写对应接口的实现类：

import java.util.List;

import java.util.regex.Pattern;

import org.springframework.data.document.mongodb.MongoTemplate;

import org.springframework.data.document.mongodb.query.Criteria;

import org.springframework.data.document.mongodb.query.Query;

import org.springframework.data.document.mongodb.query.Update;

import com.mongo.entity.Person;

import com.mongo.intf.AbstractRepository;

public class PersonRepository implements AbstractRepository{

private MongoTemplate mongoTemplate;

@Override

public List<Person> findAll() {

return getMongoTemplate().find(new Query(), Person.class);

}

@Override

public void findAndModify(String id) {

getMongoTemplate().updateFirst(new Query(Criteria.where("id").is(id)), new Update().inc("age", 3));

}

@Override

public List<Person> findByRegex(String regex) {

Pattern pattern = Pattern.compile(regex,Pattern.CASE\_INSENSITIVE);

Criteria criteria = new Criteria("name").regex(pattern.toString()); return getMongoTemplate().find(new Query(criteria), Person.class); }

@Override

public Person findOne(String id) {

return getMongoTemplate().findOne(new Query(Criteria.where("id").is(id)), Person.class);

}

@Override

public void insert(Person person) {

getMongoTemplate().insert(person);

}

@Override

public void removeAll() {

List<Person> list = this.findAll();

if(list != null){

for(Person person : list){

getMongoTemplate().remove(person);

}

}

}

@Override

public void removeOne(String id){

Criteria criteria = Criteria.where("id").in(id);

if(criteria == null){

Query query = new Query(criteria);

if(query != null && getMongoTemplate().findOne(query, Person.class) != null)

getMongoTemplate().remove(getMongoTemplate().findOne(query, Person.class)); }

}

public MongoTemplate getMongoTemplate() {

return mongoTemplate;

}

public void setMongoTemplate(MongoTemplate mongoTemplate) {

this.mongoTemplate = mongoTemplate;

}

}

5.实现类注入到service层调用

## Webservice + httpclient

我在做个人信息的时候涉及到了给用户下发短信的一个业务，当时调用的是一家公司的webservice短信接口，webservice有三种实现框架CXF,Xfire,axis2,用的最多的也是CXF因为他是可以和spring进行整合的，为了保证webservice的安全性我们采用了基于WS-Security标准的安全验证(使用CXF回调函数，WS-Security主要涉及三个反面，身份验证，签名和加密，通过<jaxws:outinterceptors>拦截器的方式设置用户名密码，签名就主要防止消息在传输过程中被进行攻击，加密就是对soap包加密)。首先通过wsdl2java根据发布的webservice服务端地址的wsdl生成客户端调用的中间桥梁java类，将生成的java类拷贝到客户端项目中，配置spring-client.xml文件，通过jaxws:client定义一个bean,并通过address属性指明要访问的webservice的服务地址，通过serviceClass指明充当中间桥梁的服务类，之后获取该bean,就可以通过它结合服务端给我们提供的开发文档来访问发布的webservice接口中的方法。

HttpClient---顾名思义，这是http协议的一个客户端封装包，它将http协议的通信规范进行了包装，提供了一些通用的API来进行基于http协议的通信的编程，基本上我们在浏览器上进行的访问http网站的操作（浏览器本身就是一个富客户端），都可以使用httpClient提供的API来代替。在编程中我们一般是基于它编程来代替“人”访问http网站的操作。

WebSevice---是基于SOAP协议，它的目的是：使用开放的XML（标准通用标记语言下的一个子集）标准，使得运行在不同机器上的不同应用无须借助附加的、专门的第三方软件或硬件， 就可相互交换数据或集成。它的核心是解决不同应用间的数据交换，因为SOAP使用XML消息调用远程方法，这样web services可以通过HTTP协议的post和get方法与远程机器交互。

http和soap协议的区别。http速度效率更高，soap能处理复杂的数据。

WebService和httpcllient 区别。：（这个必背）

HttpClient对认证机制提供了全面的支持。用户认证过程需要一组用户名/密码进行认证用户身份。HttpClient附带多个AuthScheme的实现。 AuthScheme接口表示一个抽象的面向挑战-应答的认证机制。解析和处理目标服务器发送的挑战并且对受保护资源的请求做出应答。

WebService一般通过iis进行匿名调用。这种方式不安全。我们可以使用CXF的拦截器通过自定义的Interceptor，可以改变请求和响应的一些消息处理，其中最基本的原理还是一个动态代理。设置入拦截器。来验证用户的账号和密码。

**服务端的配置**

首先要导入cxf的jar包

然后在web.xml中配置cxf的webservice服务

发布一个接口并添加@WebService注解，同时也要在实现类上添加@WebService注解 并说明实现了哪个接口，之后在spring-webservice.xml中发布webService服务

通过jaxws:endpoint配置implementor和address来表明实现服务的类，以及发布的地址，

最后在浏览器中输入相关的webservice地址?wsdl来验证服务是否发布成 功。

**webservice客户端的配置**

首先通过wsdl2java根据发布的webservice服务端地址的wsdl生成客户端调用的中间桥梁java类，将生成的java类拷贝到客户端项目中，

配置spring-client.xml文件，通过jaxws:client定义一个bean,并通过address属性指明要访问的webservice的服务地址，通过serviceClass指明充当中间桥梁的服务类，

之后获取该bean,就可以通过它来访问发布的webservice接口中的方法。

WebService安全性浅谈

第一、禁止匿名访问，对WebService进行认证，这步主要是通过操作系统的安全机制来实现的，也就是只有满足一定要求的帐号才能访问到服务器，才能访问IIS；

第二、在第一种方法的基础上对WebService里的方法进行加密，比如有个WebService方法是根据顾客ID获取数据库中的顾客的详细资料为GetCustomerDetailByCustomerID(string custID);如果只提供一个参数，则很容易被别人访问调用，因此我们对这个方法进行加密GetCustomerDetailByCustomerID(string scustID,string custID,ecustID);这样，只有提供正确的scustID与ecustID这二个参数才能成功调用这个方法，而对于这二个参数scustID与ecustID，则可以通过加密方法生成一个字符串，也可以写一个加密的工具类。

## 服务器调优

其实每个公司都关心的问题,根据不同的情况去采用不同的解决方法,像之前我们在做项目时，考虑到服务器性能的问题，最开始想到使用纵向扩展，来增加硬件的配置提高其性能，但这样做比较耗费资金，而且服务器内存空间也是有限的；所以后来就使用横向扩展来达到这一目的

当时我们使用nginx+tomcat进行负载均衡，在我们不进行负载均衡之前，那所有的请求都由一台tomcat进行处理，这样会使我们的tomcat所承受的压力增大，而我们进行负载均衡之后，同样数量的请求经过nginx将其分发到多台tomcat进行处理，从而降低每台tomcat所承受的压力,而且当其中一台机器宕机时，其他机器还可以继续提供服务，保证服务不间断。

## Redis解决session共享

当时项目在部署完成后，遇到一个问题，用户登录输入验证码的时候，明明验证码输入的正确，但总是提醒说验证码不正确从而不能正常登录，经过分析后发现有可能第一次请求被发送到t1上，那么放在session中的验证码就被放到了t1上，当用户输入验证码点击登录时，新发送的请求有可能被发送到t2上面，这样在进行对比时就肯定会不一致从而提示验证码输入错误，后来我就考虑使用ip\_hash这种负载均衡策略来代替默认的轮询策略，虽然解决了验证码错误问题，但是在后续的测试中发现如果用户在使用过程中突然一台服务器宕机了，那么因为session在这台服务器上存储着，所以就会提示用户重新登录，这样使用户的体验度非常不好，最后就通过将session信息保存到redis服务器中从而在多台web服务器中实现session共享，这样就解决了上面所说的那些问题。

nginx+tomcat负载均衡的配置(被动说)  
1.首先在nginx的conf文件夹下找到nginx.conf文件  
2.其次在该文件中的http标记下添加upstream backend并配置要转发到的多台tomcat及其端口号，设置负载均衡策略（当时我采用的是基于weight的负载均衡策略）

4.最后再在各台tomcat中找到server.xml配置文件，更改Connector标签中的端口号保持和nginx配置文件中tomcat的端口号一致即可。

其实在这几年做项目中，我感觉难的点就是性能优化以及安全方面的问题，我自己也进行了一些研究和总结，比如sql优化,缓存技术。

当时数据库用了3台mysql采用读写分离来同步。前端网站和后端运营中心是两个独立的项目。两个项目指向同一数据库。

[MySQL](http://lib.csdn.net/base/mysql" \t "http://blog.csdn.net/jaryle/article/details/_blank" \o "MySQL知识库)的读写分离的基本原理：

让master（主**[数据库](http://lib.csdn.net/base/mysql" \t "http://blog.csdn.net/jaryle/article/details/_blank" \o "MySQL知识库)**）来响应事务性操作，让slave（从数据库）来响应select非事务性操作，然后再采用主从复制来把master上的事务性操作同步到slave数据库中。

## poi

POI主要是对office文件的读写控制，通过程序解析文件中的内容，或者将数据内容生成文件导出，主要就是当我们通过条件查询出相关的数据，这时候点击导出按钮，把相关查询参数传到action层，调用service层和dao层查询出相关的数据信息，返回给action层，然后调用POI的API方法，通过HSSFWorkBook创建工作薄，然后创建sheet页，创建行。通过行创建单元格。最后通过cell的setValue方法给单元格赋值。当然，可以根据需要通过Style来实现设置单元格格式。包括边框、背景色高亮等。做数据导出的时候，因为2003版本的excel，最大行数是65535行，所以如果数据比较多的时候，就做多个sheet页。

导入数据，当我们点击导出按钮时，首先弹出一个dialog框，选择对应的excel文件，上传到文件服务器，然后点击开始导入按钮，调用action层，使用poi对excel进行解析，首先通过上传文件的路径，从服务器上获得这个文件，考虑到客户上传的excel文件版本不好统一，所以我们通过上传的excel后缀判断属于03版本，或者07版本，因此来判断使用的解析类HSSFWORKBOOK，XSSFWORKBOOK，来获得对应的工作簿解决这个问题，再通过getNumberOfSheets获得所有的sheet页，循环sheet页，循环里面的行，循环里面的格将获得的值放入对象，对象放入集合，传到service层，再传到数据库

## nginx+Tomcat 集群介绍，轮询，session共享如何实现

(反向代理的解释：反向代理（Reverse Proxy）方式是指以代理服务器来接受internet上的连接请求，然后将请求转发给内部网络上的服务器，并将从服务器上得到的结果返回给internet上请求连接的客户端，此时代理服务器对外就表现为一个服务器。)

重点两部分：一、负载均衡二、tomcat集群

所谓tomcat集群，就是可以向外提供并行服务的多台机器，任何一台服务器宕机，其它服务器可以替代它向外提供服务，而不影响用户访问。

nginx是一个常用的反向代理服务，可自定义模块，实现请求转发及负载均衡（根具体采用策略有关）。为了tomcat集群的高可用性，还需要实现nginx的双机热备。

一，如果仅是对外提供一个页面访问，不用区分单一用户（不区分每个访问session，不涉及用户权限，用户资料等内容），仅仅配置nginx负载均衡策略即可。

nginx负载均衡策略主要分一下四种：

1)、轮询（默认）

每个请求按时间顺序逐一分配到不同的后端服务器，如果后端服务器宕机，能自动剔除。

2)、ip\_hash 每个请求按访问ip的hash结果分配，这样每个访客固定访问一个后端服务器。

3)、fair 按后端服务器的响应时间来分配请求，响应时间短的优先分配。

4)、url\_hash 按访问url的hash结果来分配请求，使每个url定向到同一个后端服务器，后端服务器为缓存时比较有效。

二，如果涉及到用户session，做一些鉴权缓存、存放临时信息时，就必须做tomcat的session共享。

目前可参考到的session共享方式主要分为两种。

1）利用tomcat自带的组播机制，实现session复制。

对tomcat及应用的若干配置文件进行配置即可实现，网上有很多资料可参考。但这种方式些弊端，看过一些资料，不建议用session复制的方式。在实际使用过程中，也发现有存在session莫名失踪的现象。

2）利用第三方机制存储session。

比较常见的是tomcat集成memcached服务器来存储session。实际项目中，我们采用过利用redis实现session存储，redis高效的存取性能为高效的访问提供了保障，但是目前redis的集群功能似乎没有发布，如何解决redis的单点故障需要研究。

小结：是否实现session共享与nginx的负载策略有很大关系。比如采用轮询策略，就必须实现session共享，因为客户端会访问到每台服务器；而如果采用ip\_hash策略，就可以不用考虑session共享的问题了，但是ip\_hash有些缺陷使它不能随便使用（如多台pc使用同一个外网ip）。

最近发现一个nginx的粘连模块（类似session粘连），可以看做nginx的第5种均衡策略。它利用客户端cookie，对其写入一个route参数，每次访问可以根据route的值，固定的访问一台服务器，解决的session共享的问题

四、总结

　谁能想到实现一个高性能的负载均衡集群会如此简单。Nginx的功能如此强大，配置却如此简单，我们还有什么理由拒绝它呢？这比我们动不动就十多万至几十万人民币的F5 BIG-IP、NetScaler等硬件负载均衡交换机廉价了不知多少。此外，大家别忘了Nginx不仅仅是一个反向代理服务器，它本身也可以托管网站，作为Web服务器，进行Http服务处理。

# 数据库篇

## 数据库的主从同步读写分离:

当时我们给数据库也做了优化，配置了主从同步和通过mycat进行读写 分离；MYSQL主从同步架构是目前使用最多的数据库架构之一，尤其是负载比较大的网站。  
原理是：从服务器的IO线程到主服务器获取二进制日志，并在本地保存为中继日志，然后通过SQL线程来在从库上执行中继日志中的内容，从而使从库和主库保持一致。配置好主从同步之后我们还通过mycat来配置了读写分离；读写分离：数据分布,负载均衡,高可用性和容错,备份.  
使用：就是我们搭建两台数据库服务器，一台作为主服务器master，一台作为从服务器slave,  
1.要更改mysql数据库中的my-defult.ini文件，将文件名改为my.ini,然后更改里面的配置，包括数据库路径，端口号。编码集、log-bin文件名，开放的数据库等信息。  
2.在主数据库，新增一个用户，并指定replication权限  
3.在主数据库里面运行show master status;记下file和position字段对应的参数。  
4.在从库设置它的master  
5. stop slave; #停止服务再输入#指定用户名密码，将里面的master\_log\_file和master\_log\_pos对应刚才show master status记下的参数  
6.开启服务，start slave   
7. 通过show slave status来查看同步的状态  
这样就实现了主从同步。

## 主从同库延迟问题？

1.主库和同库尽可能在同一个局域网内，交换机网卡采用千兆网卡。  
2.主数据库更新完成之后产生的操作日志不是瞬间产生的，我们可以通过设置sync\_binlog=1, 让它瞬间产生磁盘日志，（n=1指主数据库只要操作一次，就产生一次磁盘日志，n=10，就是操作10次，产生一次），从数据库可以依赖磁盘日志，瞬间产生同步，可以达到减低延迟的效果。  
3. 设置主库和从库读取日志失败之后，及时重新建立连接，延迟缩短。

## Sql优化:

避免在 where 子句中对有索引的字段进行运算,这会导致索引失效，从而进行全表扫描。外键必须加索引

在 where 及 order by 涉及的列上建立索引，要尽量避免全表扫描。

在设计表时要避免表中字段出现null的情况，通常要为其设置默认值。

避免在查找时放弃使用索引而进行全表扫描。

Sql语句字段尽量大写。

SELECT语句中避免使用'\*’，只查询需要返回的字段 ，这样可以减少oracle解析sql语句的时间。

## 索引、存储过程、函数语法及各自的优缺点：

索引

索引是表的索引目录，在查找内容之前先查目录中查找索引位置，从而快速定位查询数据；

a、普通索引

# 创建表 + 索引

create table in1(

nid int not null auto\_increment primary key,

name varchar(32) not null,

email varchar(64) not null,

extra text,

index ix\_name (name)

)

# 创建索引

create index index\_name on table\_name(column\_name)

# 删除索引

drop index\_name on table\_name;

# 查看索引

show index from table\_name;

#注意：对于创建索引时如果是BLOB 和 TEXT 类型，必须指定length。

create index ix\_extra on in1(extra(32));

如何正确使用索引

# like '%xx'，避免%\_写在开头

select \* from tb1 where name like '%n';

存储过程：

存储过程是一个SQL语句集合，类似函数，需要主动调用。

1创建存储过程

# 无参数存储过程

# 创建存储过程

delimiter //

create procedure p1()

BEGIN

select \* from t1;

END//

delimiter ;

# 执行存储过程

call p1()

都说了类似函数，那必须得可以接收参数，且参数有三类：

in 仅用于传入参数用

out 仅用于返回值用

inout 既可以传入又可以当作返回值

# 有参数存储过程

# 创建存储过程

delimiter \\ # 结尾分号改为\\

create procedure p1(

in i1 int,

in i2 int,

inout i3 int,

out r1 int

)

BEGIN

DECLARE temp1 int; # 创建申明局部变量

DECLARE temp2 int default 0;

set temp1 = 1;

set r1 = i1 + i2 + temp1 + temp2;

set i3 = i3 + 100;

end\\

delimiter ;

# 执行存储过程

DECLARE @t1 INT default 3;

DECLARE @t2 INT;

CALL p1 (1, 2 ,@t1, @t2);

SELECT @t1,@t2;

函数

函数，与存储过程不同的是有return值。

1自定义函数

delimiter \\

create function f1(

i1 int,

i2 int)

returns int

BEGIN

declare num int;

set num = i1 + i2;

return(num);

END \\

delimiter ;

索引优缺点：

优点： 1.创建唯一索引，保证数据库表中每一行数据的唯一性

2.大大加快数据的检索速度，这也是创建索引的最主要原因

3.减少磁盘IO(像字典一样可以直接定位)

缺点: 1.创建索引和维护索引要耗费时间，这种时间随着数据量的增加而增加

2.索引需要占用额外的物理空间

3.当对表中的数据进行增加、删除和修改的时候，索引也要动态维护，降低了数据的维护速度

存储过程 优缺点

优点： 1.简化了复杂的业务逻辑，根据需要可以重复使用

2.屏蔽了底层细节，不暴露表结构即可完成操作

3.降低网络通信量，多条语句可以封装成一个存储过程来执行

4.设置访问权限来提高安全性

5.提高执行效率，因为它是预编译的以及存储在数据库中

缺点； 1.可移植性差，相同的存储过程并不能跨多个数据库操作

2.大量使用存储过程后，首先会造成服务器压力大，而且维护难度页逐渐增加

函数 优缺点:

优点： 1.函数允许标准组件式编程，提高了SQL语句的重用性、共享性和可移植性。

2.函数可以被作为一种安全机制来利用。

3.函数能够实现较快的执行速度，能够减少网络流量。

缺点： 1.函数的编写比单句SQL语句复杂。

2.在编写函数时，需要创建这些数据库对象的权限。

## 分区和分表的区别：

分表就是按照一定的规则把一张大表给分成多张小表，在数据库中看到的是几张不同的表，在硬盘上也是不同的文件，因此在进行增删改查操作的时候要根据同样的规则去找到具体操作的表。

分区是按照一定的规则把一张大表分成不同的区块，在数据库中看到的还是一张表，只不过在硬盘上存储的时候分成了几个不同的文件。这样在进行增删改查操作的时候就像操作一张普通的表一样简单方便。

他们都是为了处理单张表的大数据量问题，都能提高性能。

## 防sql注入

针对防sql注入，我们通常是这样做的：

首先在前台页面对用户输入信息进行js验证，对一些特殊字符进行屏蔽， 比如：or ,单引号，--，= ，还有就是限制用户名输入的长度，我们一般将其限制在6---13位。另外，对于用户的敏感信息我们进行Md5加密，还有，为了增加用户体验度和用户友好度，为了不使用户看到一些详细的异常信息我们会进行错误信息页面的定制，像404,500错误。另一个我层面讲，这样做也是为了保护我们的一些重要信息。此外，我们会给特定的人分配定定的权限，而不是给其分配管理员权限！

## 数据库连接池：

数据库连接池的优点运行原理:

在我们不使用数据库连接池的时候，每次访问数据库都需要创建连接，

使用完成之后需要释放关闭连接，而这样是很耗费资源的。当我们使用

数据库连接池的时候，在tomcat启动的时候就创建了指定数量的连接，

之后当我们程序使用的时候就直接从连接池里面取，而不需要创建，同理，

当我们使用完的时候也不需要关闭连接，而是将连接返回到连接池中，供其他请求继续使用。DBCP：比较稳定。C3P0: 性能比较高。

## 索引概述

索引的优点

　1.创建唯一性索引，保证数据库表中每一行数据的唯一性

　2.大大加快数据的检索速度

　　 3.减少磁盘IO（向字典一样可以直接定位）

索引的缺点

1.创建索引和维护索引要耗费时间，这种时间随着数据量的增加而增加

2.索引需要占用额外的物理空间

3.当对表中的数据进行增加、删除和修改的时候，索引也要动态的维护，降低了数据的维护速度

## 数据库三范式：

第一范式（1NF）：强调的是列的原子性，即列不能够再分成其他几列。

第二范式（2NF）: 首先是满足第一范式，另外包含两部分内容，一是表必须有一个主键，二是没有包含在主键中的列必须完全依赖于主键，而不是部分依赖。

第三范式（3NF）: 首先满足第二范式，非主键列直接依赖于主键，消除传递依赖。

## Oracle存储过程基本语法 存储过程

**Oracle的存储过程：**

存储过程就是封装一些sql的集合，也就是一条条的sql语句，过程的优点就是简化了SQL命令加上他是预编译的，所以他的执行效率相对较高。再者如果不调用过程的话就会和数据库发生多次交互，调用过程只需要传达一个命令多有的那些执行业务逻辑都在数据库端执行，所以它降低了网络的通信量，其次存储过程大大提高了安全性，这就是优点。

缺点就是不同的数据库对过程支持的关键字都是不一样的，所以它的移植性是非常差的，再者他的维护难度也比较大，因为他没有专业的调试和维护工具，所以维护起来比较麻烦，这就是存储过程的基本概述。

　　create or replace procedure 存储过程名

　 is

　 begin

　 null;

　 end;

行1: create or replace procedure 是一个SQL语句通知Oracle数据库去创建一个叫做skeleton存储过程, 如果存在就覆盖它;

行2: IS关键词表明后面将跟随一个PL/SQL体。

行3: begin关键词表明PL/SQL体的开始。

行4: null pl/sql语句表明什么事都不做，这句不能删去，因为PL/SQL体中至少需要有一句;

行5: END关键词表明PL/SQL体的结束

## 登录注册：

关于登录，我们当时考虑到以后用户量增大情况，我们队数据库中的表进行了分表。从来来提高查询的效率，这个分表的方法可以是时间，id等；

用户注册，首先需要用户手机号的验证，这时用户填写手机号后，我们会先将用户手机号和验证码保存在缓存中，并且是要设置获过期时间，一般为15分钟，然后调用短信接口将随机生成的验证码发送给用户，当用户输入验证码点击提交时，我们会根据用户手机号去缓存中进行查询，如果验证码正确，才能跳转到填写账号信息页面 ，进行账号的填写；这是用户填写用户名时，在失去焦点是我们会进行一个验证。查看数据库中是否存在这个用户名，如果存在就提示该账户以存在，请更换用户名，还有两次的密码都必须一致可以，

登录时用户可以根据手机号，用户名，邮箱进行登录，首先因为这些都是具有唯一性的，不会存在重复的情况，所以我们都可用来登录；只要有一个条件错误，我们就会告诉用户，用户名和密码不匹配，并且提醒用户是忘记密码或者忘记用户名，然后可以进行找回；

除了正常的登录注册外，当时我们还想到了单账号登录这个功能；具体原理是：当用户登录之后，我们会以当前用户信息(唯一标示id，code等)为键，而sessionid为值，来存放在缓存中，只要浏览器不关闭，或者不跨浏览器和电脑，sessionid是相同的；

除此之外，我们还会以sessionid为键，用户信息（全部信息）为值也存放在redis中；这个是为了拦截器的使用，springmvc的拦截器是在每次进入控制层之前执行一次方法，在这个方法中，我们通过session去redis中查，如果存在对象，说明是在登录状态，我们就会让其执行方法(return true )，如果不存在，还要走一些我们需要拦截的方法，我们就会让其返回到登录页面(return false)；

如果现在在别的地方也登录同一账户，我们首先会先获取当前sessionid ，第二我们会通过用户信息(唯一标示id，code等)去redis中获取以前存放的sessionid，如果两次sessionid相同说明在同一浏览器，如果不相同，说明在不同的浏览器；我们就会删除掉原来在redis中存放的数据，将新的数据存放进去，这样原来浏览器在执行其他操作的时候就会被拦截到登录页面。

## 使用aop实现mongodb日志存储，请写出实现思路和伪代码（500字）

1).在Spring中引入AOP功能

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>${srping.version}</version>

</dependency>

<dependency>

<groupId>org.aspectj</groupId>

<artifactId>aspectjweaver</artifactId>

<version>1.7.2</version>

</dependency>

2).实现Web层的日志切面

1. 实现AOP的切面主要有以下几个要素：

● 使用 @Aspect 注解将一个java类定义为切面类。

● 使用 @Pointcut 定义一个切入点，可以是一个规则表达式，比如下例中某个package下的所有函数，也可以是一个注解等。

● 根据需要在切入点不同位置的切入内容

● 使用 @Before 在切入点开始处切入内容

● 使用 @After 在切入点结尾处切入内容

● 使用 @AfterReturning 在切入点return内容之后切入内容（可以用来对处理返回值做一些加工处理）

● 使用 @Around 在切入点前后切入内容，并自己控制何时执行切入点自身的内容

● 使用 @AfterThrowing 用来处理当切入内容部分抛出异常之后的处理逻辑

2.实现Web层的日志切面

# 电商系列问题：

项目二：**深圳网上天虹商城（ 2015/10 - 2016/9）[开发团队：10人]**

责任描述：参与设计数据库以及框架的选型。后期负责商品模块（商品类别管理（Redis），商品品牌以及商品的详情页（freemarker静态化）），报表的统计（highcharts，poi），用户管理和相关接口（webService:CXF）等模块的开发，最后配合测试部门进行相应模块的调整。

项目简介：不同于市场上大众化的电商系统平台，**天虹商城**实施“聚焦”战略，采用实体店与网上商城相结合，城乡互通的方式，提供了简易的操作，丰富的功能和完善的权限管理，涉及水果蔬菜，海鲜及日用食品销售的电子商务平台。项目分为前台访问系统和后台管理系统，采用dubbo+zookeeper分布式框架，使用Lucene来进行商品搜索，使用highcharts实现报表的统计，页面采用了FreeMarker模块技术。后台模块采用SpringMVC+MyBatis框架，前台与后台相交互，前台页面使用jQuery展示并使用Oscache缓存减少与数据库访问。后台管理员对产品类别和产品进行管理后，前台用户通过购物车进行网上选购产品，并生成订单，期间可以修改订单信息，再由后台根据产品的发货情况对订单进行状态设置。

功能模块：前台：个人中心、产品搜索、商品留言、积分兑换、支付方式、商品促销、登录、购物车、会员中心等模块。

后台：商品管理、用户管理、公告管理、留言管理、订单管理、促销管理、报表统计、物流管理、供货商维护、权限管理等模块。

在做项目时主要考虑到以下这几个问题:

第一就是互联网项目，访问量大，高并发，单个数据库压力较大。

第二就是网站的抢购，秒杀后续可能会上，架构自身要能接受这样的瞬间递增访问量。

第三就是大量的留言日志，包括后续用户注册量达到千万级，单一数据库表性能会降低。

## 产品详情页（freemarker）

对于产品详情页来说因为用户的访问量比较大，所以考虑到性能，我们会通过spring的定时器结合freemarker模板引擎将产品信息进行静态化，这样就可以防止因为访问频繁而造成的服务器压力过大问题。然后用查到数据库的商品详情数据调用freemaker的封装方法就可以了生成静态页面，只需要传递3各参数就可以了，一模板名称，二要展示的数据对象，三 输出页面的位置 freemaker，并且我们会根据产品的分类和热销程度等信息采用不同的更新策略，对于需要及时更新我们会将spring定时器改为每30分钟就执行一次，对于不太重要的我们会在每天晚上凌晨2点钟对产品信息进行重新发布。也可以当运营中心添加商品，可以在业务逻辑层增加数据到数据库之后，通过AOP后置通知传递商品ID触发网站的生成详情页的方法生成详情页。

## 静态化介绍

**我们在静态化的时候，主要涉及到网页的单页页面，访问量较大，更新少，我们可以将该页面用freemaker（velocity）生成静态化页面，这个时候访问的时候，只需要访问静态化页面，不需要访问数据库，大大降低了数据库的压力，像我们招聘网的公司详情，岗位详情（app里面的商品详情，还有app里面的新闻详情，都用freemaker生成的静态化页面），这个生成的静态化页面当时涉及到一个同步问题，数据库的数据已经更新，静态化没有更新，我们的处理方式，当新增或修改一条记录到数据库的同时，在业务逻辑层连着调用freemaker封装方法，重新生成静态化页面，将当前记录的表名+id作为生成静态化页面的名称，以此来防止静态化页面相互覆盖，我们的freemaker模板工具里封装当时是传三个参数，第一，传递模板名称，第二，传递当前的数据对象，第三，指****定生成的静态化页面存储的位置。**

**Freemarker静态化空值问题如何解决？**

**空值处理运算符**

**FreeMarker的变量必须赋值，否则就会抛出异常。而对于FreeMarker来说，null值和不存在的变量是完全一样的，因为FreeMarker无法理解null值。  
FreeMarker提供两个运算符来避免空值：  
1. !： 指定缺失变量的默认值；  
2. ??：判断变量是否存在。  
!运算符有两种用法：variable!或variable!defaultValue。第一种用法不给变量指定默认值，表明默认值是空字符串、长度为0的集合、或长度为0的Map对象。  
使用!运算符指定默认值并不要求默认值的类型和变量类型相同。**

|  |
| --- |
| **测试空值处理：**  **<#-- ${sss} 没有定义这个变量，会报异常！ -->**  **${sss!} <#--没有定义这个变量，默认值是空字符串！ -->**  **${sss!"abc"} <#--没有定义这个变量，默认值是字符串abc！ -->** |

**??运算符返回布尔值，如：variable??，如果变量存在，返回true，否则返回false。**

****

**Freemarker静态化比jsp的好处？**

**1、类加载没有 PermGen 问题**

**如果你已经开发Java Web应用程序一段时间，那么对于 JVM 的 PermGen 问题可能并不陌生。由于 FreeMarker 模板不编译成类，它们不占用 PermGen 空间，并不需要一个新的类加载器加载**

**2、模板加载器**

**直接从数据源加载页面和模板岂不是很好？也许从 CMS 或数据库。也许你只想把它们放在一个地方，可以不重新部署整个应用程序就能更新它们。那么在 JSP 中你是很难做到这一点的，但 FreeMarker 提供的模板加载器就是为了 这个目的。你可以使用内建类或者创建你自己的实现**

**3、可以在运行时嵌入模板**

**FreeMarker 能让你创建真正的模板，而不只是片段 ，还记得 JSP 中的 header 和 footer 吗？FreeMarker 允许你使用一个模板（在本例中为 head.ftl）**

**4、没有导入**

**JSP 要求你导入每个你需要使用的类，就像一个常规的 Java 类一样。FreeMarker 模板，嗯，仅仅是模板。可以被包括在另一个模板中，但目前还不需要导入类**

**5、支持 JSP 标签**

**使用 Jsp 的一个理由是有可用性很好的标签库。好消息是 FreeMarker 支持 JSP 标签。坏消息是它们使用 FreeMarker 的语法，不是 JSP 语法。**

**6、表达式语言中的方法调用**

**除非你的目标是 Servlet 3.0/El 2.2 标准的容器，那么表达式语言中方法调用是不可用的。不是所有人都同意 EL 表达式中方法调用是一件好事，但是当你需要它们的时候，使用 JSP 真的太痛苦了。 但是 FreeMarker 同等对待其中每个引用。**

**7. 内置空字符串处理**

**FreeMarker 和 Jsp 都可以在表达式语言中处理空值，但 FreeMarker 在可用性上更先进一些**

**8、共享变量**

**FreeMarker 的共享变量是我最喜欢的“隐藏”功能之一。此功能可以让你设置自动添加到所有模板的值。 例如，可以设置应用程序的名称作为共享变量。**

**9、支持 JSON**

**FreeMarker 内置 JSON 支持。 比方说你有以下的 JSON 存储到变量命名 user 的字符串中**

**10、不只是为了 Web 开发**

**最后，与 JSP 不同的是FreeMarker 模板可以在 servlet 容器之外使用。可以使用它们来生成电子邮件、 配置文件、 XML 映射等。甚至可以使用它们来生成 web 页 并将它们保存在服务器端的缓存中**

**如果用户要改界面，多写几套JSP代码供用户选择不就完了吗？**

**是，这样确实也能一定程度上支持模板化。但是这种模板必须要写代码、编译，然后发布版本。而FreeMarker的很重要的一点是定义一个新的视图，不需要写代码，不需要发版本，只需要定义一套新的模板即可。**

**所以总结下来，FreeMarker在类似于淘宝店的商品展示场景下使用，会比JSP更方便。**

**因为用户要把产品放到淘宝店的展台上，展示的商品会经常发生变化，有时候要对UI做一些美化、内容展示位置的调整等等。在这种情况下，只需要使用一个新的模板，就可以把UI进行改变。而无需对淘宝的平台代码进行修改。**

## 涉及到一个静态化页面ajax异步请求的问题

页面通过ajax发送请求到handle.ashx一般处理程序，处理程序判断请求是否由页面发出并获取参数，然后初始化全局控制变量和起到控制和标识作用的变量；传递参数和SQL语句给后台方法更新数据库中访问量数据。获取访问量数据原理同理，只是SQL语句和调用的取数据的后台方法不同罢了。

## 如何解决跨域问题？

跨域一般情况下有三种解决方法。第一种就是干掉产生跨域的源头——浏览器，修改浏览器的设置就可以了，这个不建议。第二种，是生产环境下最常用的，就是用服务器设置解决。你请求自己的服务器，让自己的服务器去请求别的服务器上的资源，就规避了跨域问题。第三种，就是自己解决，用JSONP解决。最简单的方法，就还是用jQ。在jQ的官方文档里搜索jQuery.ajax()，之后页面内搜索crossDoamin，然后仔细看看就可以了。

## 支付：

（支付我们提交企业营业执照等信息，需要申请支付宝商户号和密钥，支付宝那边会给出示例接口代码，具体的我这这， 点击订单需要提交价格到我们的项目做价格校验，我们需要调用支付宝接口，传递我们的订单编号，价格，支付宝密钥，商铺号，我们项目的回调的URl给支付宝接口，完了网站页面就会支付宝页面，客户就可以完成一系列支付动作，当支付完成后，支付宝网站会调用我们发给他的我们网站的URl我们网站通过这个URl回调，获取到支付的，订单编号，流水号，价格后，进行校验存储交易日志等动作后，返回SUCCESS给银行那边银行的支付操作事物完成）。

支付安全问才说：支付的时候，很多数据都是用MD5加密过的，在一个请求先进入我们的 项目，我们会先校验，价格是否被F12 修改过完了调用支付宝接口，转到支付宝页面，后续的客户输入账号密码，或者网银转账都是在支付宝，和银行之间完成的，钱在银行数据库，支付宝的密码，在支付宝数据库他们有自己的安全措施，而且都走的是https协议都是经过加密的。我们项目再整个过程中只是通知支付宝交钱，具体的交钱动作都有支付宝，和银行系统来完成。所以安全性没问问题。及时是网络抓包工具也只是类似知道客户要付钱给谁。具体赋的过程任何操作都是支付宝，和银行的交易。所以没有安全问题。

## 商品管理模块

这个模块包括商品管理，商品分类管理，商品参数管理，商品属性管理，规格管理，品牌管理；

在数据库中涉及到的表有产品表，产品图片表，品牌表，分类表，产品参数组表，产品参数表，产品参数值表等。

通常先在商品分类管理模块中建立产品的分类，该分类是一个树形表结构，我通过全部取出数据并对取出的数据进行递归算法生成树形结构的数据，并将其数据渲染为具有层次嵌套关系的下拉列表以及表格控件从而在页面进行使用因为在后台的很多地方都使用到分类信息，而分类信息又不容易发生变化，所以我就结合oscache将分类信息进行缓存，并封装到BaseController中，从而方便在其他Controller中直接调用该方法。

建立分类后可以在商品参数管理模块中建立和指定分类绑定的参数组以及参数信息，之后在添加产品时就可以选择产品所属的分类然后就会出现和该分类绑定的参74 数组以及参数信息供公司内部业务人员填写，保存后将参数信息保存到参数值表中，普通用户可以在 产品详情页 的 商品参数 中找到。

规格管理，属性管理的操作流程和上述流程类似。

## 报表统计

(spring定时器+poi+javamail+echarts)

我在做报表统计这个模块的时候采用了highcharts这项技术，使用了柱状图，饼状图，组合图按照时间以及分类进行了各种统计（销售渠道，客户群体，客户偏好，信息来源，产品分类，产品月销量，季度销量，年销量），除此之外因为经理要求要在每月月底将指定分类的产品信息通过excel的形式发送到相关人员的邮箱中，所以我就采用了spring定时器+poi+javamail的形式完成了上述要求。

## 统计分析模块：

统计模块由 会员统计，店铺统计，销量统计等构成，其中会员统计可以按天、周、月通过highcharts的折线图对新增的会员进行统计分析，通过highcharts的饼状图对会员所在区域进行统计，并且可以根据会员的下单量，下单商品件数，下单金额分别进行统计分析计算出买家排行top10,并支持将这会员信息通过poi导出excel结合javamail发送到指定人的邮箱中。

## 购物车及订单支付（redis）：

关于购物车，我们当时考虑到怕水军来回CRUD(增读更删)购物车，对数据库造成巨大压力，当时决定采用redis来做这个购物车。

这个购物车当时想到了两种情况一种是淘宝的模式就是在用户在未登录状态下，可以查看商品，当在详情页点击“加入购物车”或者“查看购物车”的时候我们会判断用户是否登录，如果没有登录我们会弹出一个弹框，让用户先登录，成功后跳转到登录页面，并且把商品的信息传到后台，进行保存，如果用户是登录状态，直接将商品信息传到后台，进行保存，另一种就是京东不管用户是否登录都可以添加和查看购物车，先将数据全部都保存在cookie中去，然后当用户登录上去以后就会直接将cookie中的信息保存在缓存中；

我们当时有这样一个考虑，就是第一个人在查看商品添加到了购物车，但是没有登录，只是保存在了cookie中，后来第二个人也来上jd，结果第一个人的购物车商品保存在了第二个人的账户中，存在弊端，最后我们还是选用了淘宝的模式；

在保存时候我们选择的是保存到redis缓存当中，将信息传到后台之后，我们会将数据全部都存放在一个对象当中，在将对象存放在list当中，然后以用户id（唯一标示）为键，list集合为值存放在缓存当中，当用户有新的商品添加的时候，我们通过用户id查询出来原来的信息 ，将新的信息存放在list集合中去，然后将集合在放入缓存当中，但是这个过程中需要有一点注意的地方，就是当新添加的商品本来在购物中存在，这时我们就需要进行一个判断。如果存在我们只要在原来的基础上数量添加就行了；

购物车商品的删除，点击删除的时候，我们会将商品的id获取到，同时也获取到用户id，通过用户id查询出该用户的购物车商品，然后将要删除的商品从list集合当中删除掉，再保存回去；

之后当用户点击“提交订单”时会将redis中的信息取出来插入到数据库的订单表中，并清除reidis中该购物车的信息，同时将订单表中该条记录的状态设置为1（未付款），设置过期时间，在规定时间内用户可以选择进行支付，支付成功后会将订单的状态改为2，并将财付通返回的交易信息存入数据库的交易记录表中，同时调用webservice接口发送短信通知用户支付成功，此时的订单已进入后台去处理，前台再查看订单时，

如果在规定时间内，用户没有支付，我们会取消订单，这样是避免其他用户不能购买到商品，我们会根据状态显示不同的信息，比如：状态1是未支付订单，状态为2的是支付成功等待审核状态，当后台审核通过状态会改为3，当发货时状态会改为4，这时客户就可以确认收货，确认收货后状态改为5。注意:订单支付采用的是 财付通（腾讯），支付宝(淘宝)

## 每秒处理十万订单的支付系统思路

支付系统要处理每秒十万笔订单，需要的是每秒数十万的数据库更新操作（insert加update），这在任何一个独立数据库上都是不可能完成的任务，所以我们首先要做的是对订单表（简称order）进行分库与分表。

分库策略我们选择了“二叉树分库”，所谓“二叉树分库”指的是：我们在进行数据库扩容时，都是以2的倍数进行扩容。并将每个分库中都将order表拆分成10份。

如果要支持每秒处理10万订单，那每秒将至少需要生成10万个订单ID，通过数据库生成自增ID显然无法完成上述要求。所以我们只能通过内存计算获得全局唯一的订单ID。使用Twitter的Snowflake算法，实现了全局唯一ID。订单id由分库分表信息+时间戳+机器号+自增序号组成。这样只根据订单ID，我们也能快速的查询到对应的订单信息。

我们通过对order表uid维度的分库分表，实现了order表的超高并发写入与更新，并能通过uid和订单ID查询订单信息。但作为一个开放的集团支付系统，我们还需要通过业务线ID（又称商户ID，简称bid）来查询订单信息，所以我们引入了bid维度的order表集群，将uid维度的order表集群冗余一份到bid维度的order表集群中，要根据bid查询订单信息时，只需查bid维度的order表集群即可。

我们引入了消息队列进行异步数据同步，来实现数据的最终一致性。当然消息队列的各种异常也会造成数据不一致，所以我们又引入了实时监控服务，实时计算两个集群的数据差异，并进行一致性同步。

没有任何机器或服务能保证在线上稳定运行不出故障。比如某一时间，某一数据库主库宕机，这时我们将不能对该库进行读写操作，线上服务将受到影响。

所谓数据库高可用指的是：当数据库由于各种原因出现问题时，能实时或快速的恢复数据库服务并修补数据，从整个集群的角度看，就像没有出任何问题一样。需要注意的是，这里的恢复数据库服务并不一定是指修复原有数据库，也包括将服务切换到另外备用的数据库。数据库高可用的主要工作是数据库恢复与数据修补

支付系统除了最核心的支付订单表与支付流水表外，还有一些配置信息表和一些用户相关信息表。如果所有的读操作都在数据库上完成，系统性能将大打折扣，所以我们引入了数据分级机制。

我们简单的将支付系统的数据划分成了3级：

1级：订单数据和支付流水数据；这两块数据对实时性和精确性要求很高，所以不添加任何缓存，读写操作将直接操作数据库。

第2级：用户相关数据；这些数据和用户相关，具有读多写少的特征，所以我们使用redis进行缓存。

第3级：支付配置信息；这些数据和用户无关，具有数据量小，频繁读，几乎不修改的特征，所以我们使用本地内存进行缓存。

http请求在进入web集群前，会先经过一层粗细管道。入口端是粗口，入口端是粗口，我们设置最大能支持100万请求每秒，多余的请求会被直接抛弃掉。出口端是细口，我们设置给web集群10万请求每秒。剩余的90万请求会在粗细管道中排队，等待web集群处理完老的请求后，才会有新的请求从管道中出来，给web集群处理。这样web集群处理的请求数每秒永远不会超过10万，在这个负载下，集群中的各个服务都会高校运转，整个集群也不会因为暴增的请求而停止服务。

## Junit单元测试：

我在编写完自己的功能模块后，为了保证代码的准确性，一般都会使用junit进行单元测试，当时使用的是junit4这种基于注解的方式来进行单元测试。为了和spring集成获取配置的bean,通常使用 @RunWith来加载springjunit这个核心类，使用 @ContextConfiguration来加载相关的配置的文件，通过 @Resource按名字来注入具体的bean,最后在需要测试的方法上面加上

@Test 来进行单元测试。并且在编写单元测试的时候还要遵守一定的原则如：

源代码和测试代码需要分开；测试类和目标源代码的类应该位于同一个包下面，即它们的包名应该一样；测试的类名之前或之后加上Test，测试的方法名通常也以test开头。

# 框架篇：

## SpringMVC的运行原理是：

整个处理过程从一个HTTP请求开始：

1.Tomcat在启动时加载解析web.xml,找到spring mvc的前端总控制器DispatcherServlet,并且通过DispatcherServlet来加载相关的配置文件信息。

2.DispatcherServlet接收到客户端请求，找到对应HandlerMapping，根据映射规则，找到对应的处理器（Handler）。

3.调用相应处理器中的处理方法，处理该请求后，会返回一个ModelAndView。

4.DispatcherServlet根据得到的ModelAndView中的视图对象，找到一个合适的ViewResolver（视图解析器），根据视图解析器的配置，DispatcherServlet将要显示的数据传给对应的视图，最后显示给用户。

## struts2和springMVC的区别

**[架构](http://lib.csdn.net/base/architecture)1、**Struts2是类级别的拦截， 一个类对应一个request上下文，SpringMVC是方法级别的拦截，一个方法对应一个request上下文，而方法同时又跟一个url对应,所以说从本身上SpringMVC就容易实现restful url,而struts2的架构实现起来要费劲，因为Struts2中Action的一个方法可以对应一个url，而其类属性却被所有方法共享，这也就无法用注解或其他方式标识其所属方法了。

**2、**由上边原因，SpringMVC的方法之间基本上独立的，独享request response数据，请求数据通过参数获取，处理结果通过ModelMap交回给框架，方法之间不共享变量，而Struts2搞的就比较乱，虽然方法之间也是独立的，但其所有Action变量是共享的，这不会影响程序运行，却给我们编码 读程序时带来麻烦，每次来了请求就创建一个Action，一个Action对象对应一个request上下文。

**3、**由于Struts2需要针对每个request进行封装，把request，session等servlet生命周期的变量封装成一个一个Map，供给每个Action使用，并保证线程安全，所以在原则上，是比较耗费内存的。

**4、** 拦截器实现机制上，Struts2有以自己的interceptor机制，SpringMVC用的是独立的AOP方式，这样导致Struts2的配置文件量还是比SpringMVC大。

**5、**SpringMVC的入口是servlet，而Struts2是filter（这里要指出，filter和servlet是不同的。以前认为filter是servlet的一种特殊），这就导致了二者的机制不同，这里就牵涉到servlet和filter的区别了。

**6、**SpringMVC集成了Ajax，使用非常方便，只需一个注解@ResponseBody就可以实现，然后直接返回响应文本即可，而Struts2拦截器集成了Ajax，在Action中处理时一般必须安装插件或者自己写代码集成进去，使用起来也相对不方便。

**7、**SpringMVC验证支持JSR303，处理起来相对更加灵活方便，而Struts2验证比较繁琐，感觉太烦乱。

**8、** MVC和Spring是无缝的。从这个项目的管理和安全上也比Struts2高（当然Struts2也可以通过不同的目录结构和相关配置做到SpringMVC一样的效果，但是需要xml配置的地方不少）。

**9、** 设计思想上，Struts2更加符合OOP的编程思想， SpringMVC就比较谨慎，在servlet上扩展。

**10、**SpringMVC开发效率和性能高于Struts2。

**11、**SpringMVC可以认为已经100%零配置。

## hibernate缓存概述:

hibernate分为一级缓存即session缓存也叫事务级别的缓存以及

二级缓存sessionFactory即应用级别的缓存,还有查询缓存即三级缓存.

一级缓存的生命周期和session的生命周期保持一致，

hibernate默认就启用了一级缓存，

不能将其关闭，可以通过session.clear()和session.evict(object)来管理一级缓存。其中get,load,iterate都会使用一级缓存，一级缓存缓存的是对象。

二级缓存的生命周期和sessionFactory的生命周期保持一致，可以跨session,被多个session共享，hibernate3默认开启二级缓存,也可以手动开启并指定缓存插件如ehcache,oscache

等。二级缓存也只能缓存对象。

三级缓存也叫查询缓存，查询缓存是针对普通属性结果集的缓存,

对实体对象的结果集只缓存id。对query.list()起作用，query.iterate不起作用，也就是query.iterate不使用查询缓存

## Hibernate和Mybatis的区别

**两者相同点**

· Hibernate与MyBatis都可以是通过SessionFactoryBuider由XML配置文件生成SessionFactory，然后由SessionFactory 生成Session，最后由Session来开启执行事务和SQL语句。其中SessionFactoryBuider，SessionFactory，Session的生命周期都是差不多的。

· Hibernate和MyBatis都支持JDBC和JTA事务处理。

**Mybatis优势**

1 MyBatis可以进行更为细致的SQL优化，可以减少查询字段。

2 MyBatis容易掌握，而Hibernate门槛较高。

**Hibernate优势**

1 Hibernate的DAO层开发比MyBatis简单，Mybatis需要维护SQL和结果映射。

2 Hibernate对对象的维护和缓存要比MyBatis好，对增删改查的对象的维护要方便。

3 Hibernate数据库移植性很好，MyBatis的数据库移植性不好，不同的数据库需要写不同SQL。    4 Hibernate有更好的二级缓存机制，可以使用第三方缓存。MyBatis本身提供的缓存机制不佳。

[mybatis](http://www.kindve.com/forum.php?mod=viewthread&tid=23&extra=page%3D1)一个项目中用到的复杂查询基本没有，就是简单的增删改查，这样选择hibernate效率就很快了，因为基本的sql语句已经被封装好了，根本不需要你去写sql语句，这就节省了大量的时间，但是对于一个大型项目，复杂语句较多，这样再去选择hibernate就不是一个太好的选择，选择就会加快许多，而且语句的管理也比较方便。

**两者比较**

因为Hibernate对查询对象有着良好的管理机制，用户无需关心SQL。所以在使用二级缓存时如果出现脏数据，系统会报出错误并提示。

而MyBatis在这一方面，使用二级缓存时需要特别小心。如果不能完全确定数据更新操作的波及范围，避免Cache的盲目使用。否则，脏数据的出现会给系统的正常运行带来很大的隐患。

## [mybatis和ibatis的区别](http://blog.csdn.net/christgun/article/details/51456282)

1.以前用的parameterClass在mybatis中已经用不了了，mybatis里应该使用parameterType。另外resultMap里面也不能继续使用了改成了type

2.dynamic标签不能使用了

3.数据类型的声明和ibatis有了很大的差别，ibatis可以像下面这样写

#FCODE:VARCHAR2#, mybatis的话一般是这样弄的#{FMARK,jdbcType=VARCHAR}

4.mybatis现在已经没有SqlMapClient了，使用的则是SqlSession.在原来的基础上加了像selectMap,selectList,selectOne这样的方法，使用更方便了

5.加了一个叫映射器的新东西，只需要写出接口而不需要实现类就能够操作数据如

6 需要注意的是spring3.0一下的版本并不支持mybatis，

## 【$与#号的区别】

1. #将传入的数据都当成一个字符串，会对自动传入的数据加一个双引号。如：order by #user\_id#，如果传入的值是111,那么解析成sql时的值为order by "111", 如果传入的值是id，则解析成的sql为order by "id".

2. $将传入的数据直接显示生成在sql中。如：order by $user\_id$，如果传入的值是111,那么解析成sql时的值为order by user\_id,  如果传入的值是id，则解析成的sql为order by id.

3. #方式能够很大程度防止sql注入。

4.$方式无法防止Sql注入

5.$方式一般用于传入数据库对象，例如传入表名

6.一般能用#的就别用$.

## Oracle和Mysql的区别

1. MYSQL有自动增长的数据类型，插入记录时不用操作此字段，会自动获得数据值。ORACLE没有自动增长的数据类型，需要建立一个自动增长的序列号

2．单引号处理：MYSQL里可以用双引号包起字符串，ORACLE里只可以用单引号包起字符串。在插入和修改字符串前必须做单引号的替换：把所有出现的一个单引号替换成两个单引号。

3.  MYSQL处理翻页的SQL语句比较简单，用LIMIT 开始位置, 记录个数；PHP里还可以用SEEK定位到结果集的位置。ORACLE处理翻页的SQL语句就比较繁琐

4.   长字符串的处理ORACLE也有它特殊的地方。INSERT和UPDATE时最大可操作的字符串长度小于等于4000个单字节, 如果要插入更长的字符串, 请考虑字段用CLOB类型

5.  MYSQL日期字段分DATE和TIME两种，ORACLE日期字段只有DATE

6.  MYSQL的非空字段也有空的内容，ORACLE里定义了非空字段就不容许有空的内容。按MYSQL的NOT NULL来定义ORACLE表结构, 导数据的时候会产生错误。因此导数据时要对空字符进行判断，如果为NULL或空字符，需要把它改成一个空格的字符串。

7.  MYSQL里用 字段名 like '%字符串%',ORACLE里也可以用 字段名 like '%字符串%' 但这种方法不能使用索引, 速度不快，用字符串比较函数 instr(字段名,'字符串')>0 会得到更精确的查找结果。

9.  程序和函数里，操作数据库的工作完成后请注意结果集和指针的释放。

## [为什么做java的web开发我们会使用struts2，springMVC和spring这样的框架?](http://www.cnblogs.com/sharpxiajun/p/3936268.html)

　　今年我一直在思考web开发里的前后端分离的问题，到了现在也颇有点心得了， 随着这个问题的深入，再加以现在公司很多web项目的控制层的技术框架由struts2迁移到springMVC，我突然有了一个新的疑问无法得到正确的 解释，为什么我们现在做java的web开发，会选择struts2或者springMVC这样的框架，而不是使用servlet加jsp这样的技术呢？ 特别是现在我们web的前端页面都是使用velocity这样的模板语言进行开发，抛弃了jsp，这样的选择又会给我们java的web开发带来什么样的 好处，延着这个问题的思路，我又发现新的疑问，为什么现在很多java企业级开发都会去选择spring框架，spring框架给我们开发的应用带来了什 么？这么一想还真是问倒我了，我似乎很难找到一串能让人完全信服的答案，最终我发现，这些我认为我很熟悉很常用的技术，其实还有很多让我陌生不解的地方， 这些陌生和不解的地方也正是我是否能更高层次使用它们的关键，今天这篇文章我就来讲讲这些问题，不过struts2，spring这样的技术经过这么多年 的积累已经是相当庞大和复杂，它们的面很广，本人虽然已经用了它们多年，还是有很多技术不熟悉和不清楚，所以本文不是全面对我题目做出解答的文章，而是根 据我现有知识结构来理解这个问题。

　　软件里有很多优秀的框架，有一种类型的框架，它的特点是建立在一个现有技术的基础上，提供和现有技术一样业务功能的技术框架，这个新的技术框架 比原技术更加易用，更加健壮同时功能更加强大，例如：jQuery，以及本文所要谈到的struts2和springMVC，深究这些框架都是相当之复 杂，但是它们的优点其实只有一个：就是让使用者只关心核心业务的开发，框架帮你屏蔽原有技术跟业务开发无关的各类技术问题。像 jQuery，struts2或springMVC这类框架之所以优秀，就是它们在这点上做的太好了，以至于很多使用它的程序员都已经不清楚原有技术的真 实面目，因此我们要将struts2理解的更好，使用的更加熟练和深入，这里我们就要跳出struts2的技术，到struts2技术的源头 servlet，仔细研究下servlet的特点，只有这样我们才能把struts2框架学的更好。

　　Servlet的作用是接收浏览器传给服务端的请求（request），并将服务端处理完的响应（response）返回给用户的浏览器，浏览 器和服务端之间通过http协议进行沟通，其过程是浏览器根据用户的选择将相关信息按http协议报文的规范组装请求的http报文，报文通过网络传输到 指定的服务器，服务器通过特定的web容器接收这个报文信息，例如：tomcat，jetty，jboss这样的web容器，web容器会将http报文 解析出来，如果是用户请求，最终解析出来的报文信息会用一个request对象存储起来，服务端使用这个request做完相应的处理后，服务端程序将结 果信息封装到response对象里，然后将response对象交给web容器，web容器则把这个response对象转变为http协议的报文，并 将报文回传给浏览器，浏览器最后解析这个响应报文，将最终结果展示给用户。

　　Web容器创造了servlet接口，servlet接口就是开发人员自己实现业务逻辑的地方，程序员开发servlet就好比做填空题，而填 空题的语境或者说上下文提示就是由request和response对象，但是javaEE规范里的servlet接口很简单，就三个方法 init，service和destory，但是这个接口太笼统，所以规范里还提供了一个HttpServlet类，这个类根据http请求类型提供了 doGet，doPost等方法，servlet接口最大的特点就是根据http协议的特点进行定义，因此做servlet开发时候如果使用者对http 协议特点不是特别熟悉，都会碰到或多或少令人迷惑的问题，特别是碰到一些复杂特殊的请求时候：例如文件上传，返回特殊的文件格式到浏览器，这时候使用 servlet开发就不是很方便了，servlet开发还有个问题可能大家常常被忽视，就是请求的数据的类型转化，http协议传输都是文本形式，到了 web容器解析后也是文本类型，如果碰到货币，数字，日期这样的类型需要我们根据实际情况进行转化，如果页面传送的信息非常多，我们就不得不做大量类型转 化，这种工作没有什么技术含量，是个体力活而且很容易导致程序错误。同时java的企业开发都是围绕javabean进行，类型转化好的数据还要封装到对 应的javabean里，这种转来转去的事情对于项目开发绝对不是什么好事情，所以古老的struts1为这种问题找到了一种解决方案，就是定义了一个 DTO对象（数据传输对象），专门负责做这样的事情，不过到了struts2，整个替代servlet的action本身就是一个javabean。

　　Java的企业开发一个技术特点就是使用javabean进行的，struts2的特点之一就是它替代servlet的操作类就是一个典型的 javabean，首先struts2框架将页面传输的数据进行类型转化和封装后将请求信息封装到了这个javabean的属性里，这样我们开发web程 序时候就省去了烦心的类型转化和封装的问题，前面我讲到传统的servlet是根据http协议进行定义的，它会按你请求方式（post还是get方式） 来处理用户的请求，但是对于一名程序开发人员而言，一个请求，具体到一个url，其实对于服务端而言就是服务端对外提供的一个功能，或者说是服务端对外的 一个动作，如果我们使用servlet开发程序我们就得把http的动作转化为具体的业务动作，这就让程序开发变得繁琐，增强了开发的难度，所以 struts2替代servlet的javabean就屏蔽了servlet里http的请求方式和具体业务动作转化的问题，javabean里的每一个 方法都可以和每一个url请求一一对应，这必然减轻了开发的难度问题。

Servlet另一个作用就是构造response对象，让页面获得正确的响应，其实现代的浏览器是一个多媒体工具，文字，图片，视屏等等东西都可 以在浏览器里显示，资源的不同就会导致http响应报文的差别，如果我们使用servlet开发就要根据资源的不同在java程序里用硬编码的形式处理， 这样的程序很难复用，而且如果程序员对某种资源的处理理解不到位，就会导致问题的出现，struts2通过配置文件的形式将这样的逻辑从java程序里剥 离出来，使用配置的方式进行统一管理，这个做法和spring的AOP方式类似，这样就让结果处理方式更加统一，更加利于管理，同时也提升了程序的健壮性 以及降低了开发的难度。

　　Servlet在MVC开发模式里就是其中C层即控制层，控制层就像俄罗斯的双头鹰（一个头向东看一个头向西看）一样，一个头向M层模型层看， 一个头向V层视图层看，模型层也是用java编写的，控制层也属于服务端语言开发，所以M层和C层的沟通没有天然的障碍，但是和V层视图层就不一样了，这 是一个跨语言的沟通，对于浏览器，它只懂得html，javascript和css，浏览器是理解不了java这种语言的东西，但是要让服务端的东西能被 浏览器理解接受，我们就必须得把服务端的响应信息放到页面里，因此就需要一个技术把java的信息转化到html页面里，这就是javaEE规范里提供了 jsp技术，jsp其实是一种服务端技术而非客户端技术，不过它看起来似乎更像html技术，最早的jsp开发里都是直接将java代码写到页面里，这种 坏处谁都知道，之后javaEE规范提供了自定义标签技术，使用一种类似html标签的方式来解析java代码，struts2框架提供了一整套完整的自 定义标签技术，这似乎听起来不算啥，但是它的作用非凡，因为自定义标签之所以叫自定义就是每个人都可以自己来定义，如果没有一个规范必然产生混乱，而且一 套完善的自定义标签是个系统工程，一套完整的自定义标签相当于我们在自己定义一套新的开发语言，做程序的人听到这个一定就会明白开发一套完整的自定义标签 的工作量和开发难度都是难以想象的，而且自定义标签都是和控制层紧密相连，其难度又会增加一个维度，所以struts2提供的自定义标签对于业务开发带来 的将是质的飞越。

　　Servlet里还有两个重要的技术：监听器和过滤器，对于监听器在web开发里使用的场景比较少，都是一些十分特别的情况才会使用，大部分 web开发里可以忽略它的使用，我们用的最多的监听器可能就是对ServletContext创建和销毁的监听器，ServletContext是整个 web应用的全局对象，它和Web应用的生命周期绑定在一起，因此使用这个监听器对Web应用的全局信息进行初始化和销毁操作，例如spring容器的初 始化操作。比较有意思的是过滤器，在struts2里有个拦截器，它们的作用相同都是用来拦截请求的，因为拦截器是struts2的特有功能，在 struts2里使用拦截器自然比使用过滤器更顺手，其实拦截器所用的技术比过滤器更加先进，因为拦截器使用了反射技术，因此拦截器拦截的面更大，控制请 求的能力更强，它能完成的任务也会更加的丰富多彩。

　　在我第一次接触struts2时候，有人告诉我struts设计的一个目的就是想屏蔽在控制层里操作request和response对象，因 为这两个http协议的儿子会造成web开发里思路的混乱，但是我在实际开发里却经常不自觉的使用这两个对象。而且本人做前端开发非常喜欢使用ajax， 使用ajax技术时候我就很讨厌struts2的自定义标签，我更加喜欢在页面里用javascript技术处理各种信息，最终struts2在我眼里就 是一个servlet的变体，因此曾经有段时间我常常在想是不是可以抛弃struts2，直接用servlet，因为struts2里用到了太多反射机 制，特别是使用注解做配置（注解是用反射实现的），在java里反射的执行效率是非常低的，直接使用servlet一定能提升web应用的执行效率。其实 这个倒很难做到，因为当时我没法在servlet里灵活的运用spring技术。

下面我要谈谈spring了。

　　spring技术可以说是java企业开发里最重要的技术，不过真的理解spring的作用和意义还真是一件麻烦的事情，很多人对spring 理解其实都是停留在使用阶段（例如：声明式事务很好用等等），当今的spring技术生态环境里可谓是蔚为壮观，spring已经包罗万象，它的内容之多 完全不亚于它的本源java语言了，而spring这么大的框都是建立在ioc和aop技术之上，只有深入理解了这两个技术我们才能明白为什么 spring这个框能装的下那么多东西了。

　　首先是ioc，ioc技术第一个解释叫做控制反转，它还有个解释就是依赖注入，这两个名字很难从字面理解，但是当你理解它的原理后就会发现它们 的描述是何等准确。Ioc技术的本质就是构建对象的技术换句话说就是将一个类实例化成对象的技术，在java里实例化类通过new关键字进行的，每次 new一个类都会产生一个新的实例对象，这么做视乎很浪费，有时这种浪费还挺危险，因为在程序开发时候我们常常只需要某个类永远只能产生一个的实例对象这 个时候就得使用单例模式，此外在设计模式里还可以通过工厂方式产生对象，使用过spring的人看到上面的文字就知道了，spring里bean的定义就 和上面的内容一一对应，scope属性single产生单例对象，prototype产生新对象，bean还可以通过工厂方式产生对象，可以说 spring的bean就是制造对象的工具。面向对象编程里对象相当于显示生活中的一个实体，例如我们有个对象作用是完成打猎的操作，那么打猎这个对象内 部包含两个辅助对象：人和枪，只有人和枪赋予了打猎这个对象，那么打猎对象才能完成打猎的操作，但是构建一个人和枪的对象并不是看起来那么简单，这里以枪 为例，要创造一把枪我们需要金属，需要机床，需要子弹，而机床和子弹又是两个新对象，这些对象一个个相互嵌套相互关联，大伙试想下如果我们在java代码 里构建一个枪的对象那是何其的复杂，假如我们要构造的不是简单的枪对象而是更加复杂的航空母舰，那么构造这个对象的成本之高是让人难以想象的，怎么来消除 这种对象相互嵌套相互依赖的关系了？spring提供了一种方式，这种方式就是spring提供一个容器，我们在xml文件里定义各个对象的依赖关系，由 容器完成对象的构建，当我们java代码里需要使用某个实例的时候就可以从容器里获取，那么对象的构建操作就被spring容器接管，所以它被称为控制反 转，控制反转的意思就是本来属于java程序里构建对象的功能交由容器接管，依赖注入就是当程序要使用某个对象时候，容器会把它注入到程序里，这就叫做依 赖注入。在java开发里我们想使用某个类提供的功能，有两种方式，一种就是构造一个新的类，新的类继承该类，另一种方式则是将某个类定义在新类里，那么 两个类之间就建立一种关联关系，spring的ioc容器就是实现了这种关联关系（记住不是继承关系哦），那么某个类要被赋予到新类有哪些办法了？一般只 有两种：一种就是通过构造函数，一种就是通过setXXX方式，这也是spring容器使用到了两种标准的注入方式。

　　不管是上面说的继承方式，还是关联方式其实都是增强目标对象能力的开发手段，在设计模式里有一种代理模式，代理模式将继承模式和关联模式结合在 一起使用，代理模式就是继承模式和关联模式的综合体，不过这个综合体的作用倒不是解决对象注入的问题，而是为具体操作对象找到一个保姆或者是秘书，这就和 小说里的二号首长一样，这个二号首长对外代表了具体的实例对象，实例对象的入口和出口都是通过这个二号首长，因为具体的实例对象是一号首长，一号首长是要 干大事的，所以一些事务性，重复性的工作例如泡茶，安排车子，这样的工作是不用劳烦一号首长的大驾，而是二号首长帮忙解决的，这就是aop的思想，aop 解决程序开发里事务性，和核心业务无关的问题，但这些问题对于业务场景的实现是很有必要的，在实际开发里aop也是节省代码的一种方式。

　　Spring的核心技术的作用本质就是一个 沟通机制，spring总是尽全力的让沟通的双方信息畅通，同时降低双方的沟通成本，在现实机构里一个善于沟通的人肯定是该公司的领导，很会沟通的领导能 调动起各种资源的积极性，善于沟通的领导就会做到海纳百川，让各种不同人追随他，所以当今的spring就是一个大容器，什么都可以往里装。

Spring很像银行，它不能直接创造物质财富，但是一切资源都要通过它进行流通，它能控制经济发展的走向，回到程序的世界，spring的作 用是被标榜为程序之间的解耦，spring能降低不同模块之间的耦合度，原因就是在程序开发里不同模块之间信息的沟通是通过对象传递完成的，而对象能否顺 利传递就是要合理的构建好对象，而管理好对象的构建方式就能管理好对象传递，这就是spring给系统架构设计带来的好处。

# 基础方面：

## BS和CS的区别以及优缺点

C/S又称Client/Server或客户/服务器模式。服务器通常采用高性能的PC、工作站或小型机，并采用大型数据库系统，如Oracle、Sybase、Informix或 SQL Server。客户端需要安装专用的客户端软件。

B/S是Brower/Server的缩写，客户机上只要安装一个浏览器（Browser），如Netscape Navigator或InternetExplorer，服务器安装Oracle、Sybase、Informix或 SQL Server等数据库。浏览器通过Web Server 同数据库进行数据交互。 C/S的优点是能充分发挥客户端PC的处理能力，很多工作可以在客户端处理后再提交给服务器。对应的优点就是客户端响应速度快 C/S客户端的计算机电脑配置要求较高。

B/S客户端的计算机电脑配置要求较低。

C/S每一个客户端都必须安装和配置软件,客户端不必安装，使用浏览器访问，易推广。

B/S最大的优点就是可以在任何地方进行操作而不用安装任何专门的软件。

C/S每一个客户端都要升级程序。可以采用自动升级。BS客户端不必安装及维护。

C/S一般面向相对固定的用户群，程序更加注重流程，它可以对权限进行多层次校验，提供了更安全的存取模式，对信息安全的控制能力很强。一般高度机密的信息系统采用C/S结构适宜。

## 五险一金，和上交比例

五险一金的名称：养老保险，医疗保险，失业保险，工伤保险，生育险，住房公积金

比例：五险一金的标准：最低 工资的30% \* 12% 最高 工资\*12%

## HttpClient

httpClient是apache旗下的子项目适用于多项目之间的数据传输，httpClinet与webservice不同，webservice是一个重量级的框架，

适用于跨域和跨项目之间的数据传输，而httpClient是一个轻量级的框架，使用起来相对比较灵活、易用，

httpClient使用步骤，

1、创建HttpClient实例；

2、创建连接方式实例，在这里有两种连接方式：GetMethod和PostMethod

3、执行httpClient对象的excute方法；

4、读取response方法；

5、处理返回的结果，

首先判断状态码为200的说明连接成功，然后在执行 HttpEntity entity=response.getEntity();；获得相应的实体；

使用super.outString(EntityUtils.toString(entity),responsed)

将数据传输到前台

6、关闭资源，释放连接；httpclient.close()；

项目中的应用以及遇到的问题：

当时我们在做大眼睛房地产项目时候，其中一个功能模块，需要调用本公司另一个项目组中数据，当时项目经理让我去完成这个项目，

我就想到使用httpClient技术来实现这个功能，通过上网查找资料和查阅以前的笔记，

首先导入httpClient相关的jar包,通过以上6步连接要调用的项目，

开始在做添加时候出现了乱码问题，后来在postMothed中加上编码集设置 postMethod.setRequestHeader("Content-Type","application/x-www-form-urlencoded;charset=utf-8");

后解决了该问题，在文件上传时候应该注意上传的文件名，和要调用的项目中接收的文件名一致。

## 事务的隔离级别和传播特性

在数据库中,所谓事务是指一组逻辑操作单元即一组sql语句。当这个单元中的一部分操作失败,整个事务回滚，只有全部正确才完成提交。

事务的ACID属性

1. 原子性（Atomicity）

原子性是指事务是一个不可分割的工作单位，事务中的操作要么都发生，

要么都不发生。

2. 一致性（Consistency）

事务必须使数据库从一个一致性状态变换到另外一个一致性状态。(数据不被破坏)

3. 隔离性（Isolation）

事务的隔离性是指一个事务的执行不能被其他事务干扰.

4. 持久性（Durability）

持久性是指一个事务一旦被提交，

它对数据库中数据的改变就是永久性的.

在JDBC中，

事务默认是自动提交的，

每次执行一个 SQL 语句时，如果执行成功，

就会向数据库自动提交，而不能回滚

为了让多个 SQL 语句作为一个事务执行：

（1）执行语句前调用 Connection 对象的 setAutoCommit(false);

     以取消自动提交事务

（2）在所有的 SQL 语句都成功执行后，调用 commit(); 方法提交事务

（3）在出现异常时，调用 rollback(); 方法回滚事务。

五个事务隔级别分别为：lsolation的属性值

1，default

默认的事务隔离级别

2，read\_uncommitted

读未提交，一个事务可以操作另外一个未提交的事务，不能避免脏读，不可重复读，幻读，隔离级别最低，并发性 能最高

3，read\_committed

读已提交，一个事务不可以操作另外一个未提交的事务， 能防止脏读，不能避免不可重复读，幻读。

4，repeatable\_read

能够避免脏读，不可重复读，不能避免幻读

5，serializable

隔离级别最高，消耗资源最低，代价最高，能够防止脏读， 不可重复读，幻读。

七个事务的传播行为

1，propagation\_required

如果一个事务存在，则支持当前事务，如果不存在，则创建新的事务

2，propagation\_supports

如果一个事务存在，则支持当前事务，如果不存在，则非事务的方法运行

3，propagation\_mendatory

如果一个事务存在，则支持当前事务，如果存在，则抛出异常

4，propagation\_requires\_new

总是要开启一个新的事务，如果事务存在，将该事务挂起

5，propagation\_not\_supported

总是非事务方法运行，并挂起所有的事务

6，propagation\_never

总是非事务方法运行，如果事务存在则抛出异常

7，propagation\_nested

某一个事务存在，则运行在一个嵌套的事务中

## Ajax

ajax本身不支持跨域，可以通过jsonp来达到跨域目的。jsonp的本质就是通过script标签在发送url请求时多加了一个callback参数，服务端通过request获取callback参数的值也就是回调的函数名，将要传递的json数据作为参数响应给客户端进行回调。

通过eval可以将后台返回的json格式的字符串转换为json对象。

**jquery的ajax默认情况下是没有超时限制的，可以通过timeout【单位：毫秒】设置超时时间。**

## Vector & ArrayList的区别

1）  Vector的方法都是同步的(Synchronized),是线程安全的(thread-safe)，而ArrayList的方法不是，由于线程的同步必然要影响性能，因此,ArrayList的性能比Vector好。

2） 当Vector或ArrayList中的元素超过它的初始大小时,Vector会将它的容量翻倍,而ArrayList只增加50%的大小，这样,ArrayList就有利于节约内存空间。

**hashMap和hashtable的区别:**

map是一个键值对的存储接口，Map有两个具体的实现类是HashMap和HashTable,

hashmap是非线程安全的，hashtable是线程安全的，所以hashmap的效率高于hashtable;

hashmap允许空值空键，table不许空值空键；

关于map的底层，是通过数组加链表的形式来进行存放的，比如我们要存放一会key , value，我们会根据key的hashcode()值与数组的长度进行取余，通过余数放入到对应的数组位置上，然后通过key取值得时候，我们也是通过传过来的key的hashcode()与数组长度取余，然后就能很快的找到我们存放的位子。但是这样会存在一个问题，就是余数会有重复的情况，我们就加上一个链表。通过key取到链表之后，我们就将链表进行循环，然后相比较来查询出相对应的值；

**String ,stringbuffer ,stringbuilder 的区别:**

首先这些都是用来定义字符串的；

String是一个字符串常量，被final修饰过的，长度不可变，拼接字符串的时候使用+=，

每一次拼接字符串都会是一个新的内存空间；

Stringbuffer是字符串变量，存在缓存区，长度是可以变得，只需要通过append来进行追加就可以，不需要开辟新的内存空间；有synchronized 所以线程是安全；

Stringbuilder 也是字符串变量，和Stringbuffer差不多，但是没有同步锁。线程不安全；

在项目当中，我们最常用的是string ，因为使用的比较简单，不需要new对象，还提供了许多的方法；虽然stringbuffer的效率比这个要高，但是在数据量小的时候，相差不了多少，

只有在大数据量拼接的时候才会使用stringbuffer和stringbuilder;

**Set，list ,conllection,conllections的区别？**

List 和set都是接口，他们都继承与接口conllection，

List是一个有序的可以重复的集合，而set是无序的可以重复的集合；

conllection是集合的顶层接口；

conllections是一个封装了众多关于集合操作的静态化方法的工具类，因为构造方法是私有的不能实例化；

List接口中有arraylist ，linkedlist 和vector，arraylist和vector都是基于数组实现的，所以查询的时候速度快，而在进行增加删除的时候数据要比linkedlist 慢，linkedlist 是基于链式存储结构，所以在查询的速度上相对慢一点；

**抽象类和接口的区别（抽象类特点）**

抽象类通过abstract声明；

抽象类允许普通属性和普通方法存在；抽象类允许抽象方法存在；

使用abstract修饰的方法，就是抽象方法。抽象方法与普通方法的区别是：抽象方法没有方法体（没有{}）；

抽象类不能被实例化；

抽象类只能通过普通类实现后进行使用，普通类通过extends关键字来实现抽象类，并且必须实现抽象类的所有抽象方法；

**接口（interface）接口特点；**

接口通过interface声明；

接口只允许常量和抽象方法存在，常量可以省略public static final，抽象方法可以省略public abstract；

接口可以实现多继承（继承多个接口）；

接口不能被实例化；接口通过类来实现接口，实现的关键字是implements，类要实现接口的所有抽象方法，类可以实现多个接口；

抽象类也可以通过implements实现接口，并且允许不实现里面的抽象方法；

## 运行时异常与一般异常有何异同？

Java提供了两类主要的异常:运行时异常runtime exception和一般异常checked exception。checked 异常。对于后者这种异常，JAVA要求程序员对其进行catch。所以，面对这种异常不管我们是否愿意，只能自己去写一大堆catch块去处理可能的异常。

运行时异常我们可以不处理。这样的异常由虚拟机接管。出现运行时异常后，系统会把异常一直往上层抛，一直遇到处理代码。如果不对运行时异常进行处理，那么出现运行时异常之后，要么是线程中止，要么是主程序终止。

## HashMap的底层代码/原理

HashMap底层就是一个数组结构，数组中的每一项又是一个链表。

当新建一个HashMap的时候，就会初始化一个数组。

Entry就是数组中的元素，每个 Entry 其实就是一个key-value对，

它持有一个指向下一个元素的引用，这就构成了链表。

HashMap 在底层将 key-value 当成一个整体进行处理，这个整体就是一个 Entry 对象。

HashMap 底层采用一个 Entry[] 数组来保存所有的 key-value 对，

当需要存储一个 Entry 对象时，会根据hash算法来决定其在数组中的存储位置，

再根据equals方法决定其在该数组位置上的链表中的存储位置；当需要取出一个Entry时，

也会根据hash算法找到其在数组中的存储位置，

再根据equals方法从该位置上的链表中取出该Entry。

默认是构建一个初始容量为 16，负载因子为 0.75 的 HashMap。

也就是说，默认情况下，数组大小为16，那么当HashMap中元素个数超过16\*0.75=12的时候，

就把数组的大小扩展为 2\*16=32，即扩大一倍，然后重新计算每个元素在数组中的位置，

而这是一个非常消耗性能的操作，所以如果我们已经预知HashMap中元素的个数，

那么预设元素的个数能够有效的提高HashMap的性能。

**堆和栈的区别**

 堆内存用来存放由new创建的对象和数组，通过new关键字和构造器创建的对象放在堆空间，大量的对象都是放在堆空间，整个内存包括硬盘上的虚拟内存都可以被当成堆空间来使用

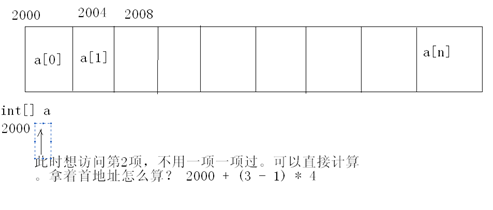
栈里面存放的是所有基本数据类型和引用数据类型，我们定义一个基本数据类型的变量，一个对象的引用，还有就是函数调用的现场保存都使用内存中的栈空间，栈空间操作最快但是也很小

**arrayList和linkedList的区别**

我们知道ArrayList是List接口的一个实现类，它的特点是查询效率高，增删效率低，线程不安全

原因是因为ArrayList底层是封装了一个数组，它是用数组实现的。

看下图，数组在内存中的存储方式：



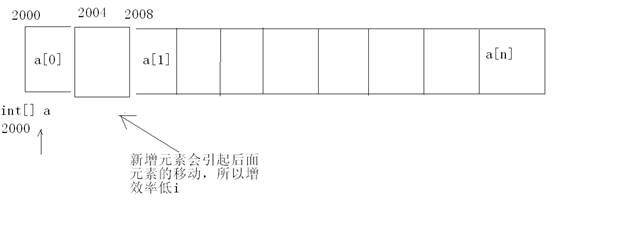
这里写图片描述

现在定义一个int[] a数组，假设它的首地址是2000，int类型占4个字节，所以a[0]的首地址是2000，a[1]的首地址就是2004，以此类推….到a[n]

所以上面的这张图，就很形象的解释了，为什么ArrayList的查询效率高，每次只需要一个偏移量就可查到

但是它的增删效率为什么会低呢？

再往下看：



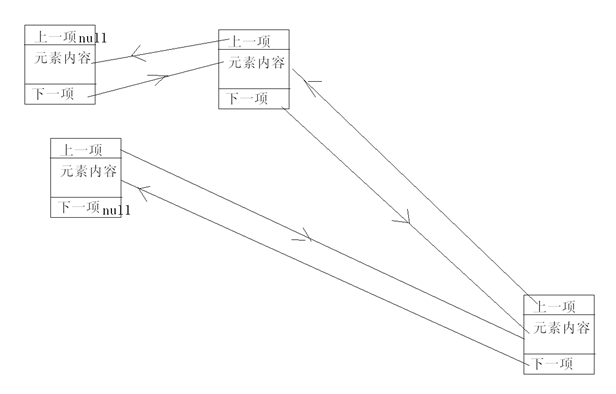
如果想在这个数组的中间或是第一项前插入一项，它的插入过程是以迭代的方式，让它后面的每一项依次往后挪，就如图上的要在第二项的位置上插入一项，其实这个过程是比较慢的，如果是你每次都在最后插入，这是个例外，因为它不用再去影响其它的元素，直接插在最后面；当然删也是同一个道理

看完了ArrayList,我们再去研究一下LinkedList

它的特点是：增删效率比较高，而查询效率低

LinkedList是底层用双向循环链表实现的List，链表的存储特点是不挨着，它存储的每个元素分为三段：上一项的地址、下一项的地址、元素的内容，而数组却没有上一项，下一项这种情况 ，因为数组只需要根据一个偏移量，就可以找到下一项或上一项

双向链表的底层结构图：



这里写图片描述

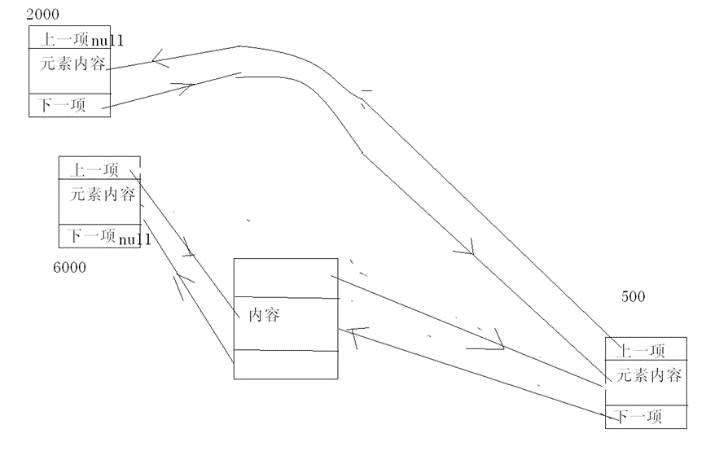
每个元素在内存中的排列像是随机的，中间没有连续性，通过地址来找到上一项或下一项，从图上应该可以理解了

那么现在问题来了，如果查询LinkedList中的某一项，肿么办？

没有好办法，只能把元素全部过一遍，这样就会比较的慢

而它的好处体现在它的增删效率非常的快，为什么呢？

看下面的图解：



假如我把右上的一个元素删掉，可以看到左上和右下的两个元素会直接连上，至于它们两个是怎么牵上手的，这里不多讲了，你懂的…………….

同理，在下面增加一个的时候，也是同一个道理，也就是说，当你增加或删除一个元素的时候，在LinkedList里，它最多只会影响两个元素，而不像ArrayList里，当在中间插入一个元素时，它后面的所有的元素都要受到影响，那么这样在一定程度上LinkedList的增删效率就会明显的高于ArrayList的

## 什么是单例多例

单例：就是所有的请求都用一个对象来处理，service和dao层的对象通常都是单例的， 之所以用单例，是因为没必要每个请求都新建一个对象，这样子   既浪费CPU又浪费内存；

单例就是该类只能返回一个实例。

单例所具备的特点：

1.私有化的构造函数

2.私有的静态的全局变量

3.公有的静态的方法

多例：指每个请求用一个新的对象来处理，比如action;  之所以用多例，是为了防止并发问题；即一个请求改变了对象的状态，此时对象又处理另一个请求，而之前请求对对象状态的改变导致了对象对另一个请求做了错误的处理；

 用单例和多例的标准只有一个， 当对象含有可改变的状态时则多例，否则单例；

## 懒汉式单例模式：

public class Singleton {

private static Singleton singleton = null;

private Singleton(){

}

synchronized public static getInstance(){

if(singleton == null){

singleton = new Singleton();

}

return singleton;

}

}

## 饿汉式单例模式

public class Singleton {

private static final Singleton singleton = new Singleton();

public Singleton(){

}

public static getInstance(){

return singleton ;

}

}

## 线程安全式单例模式：

public class Singleton{

private static Singleton singleton;

private Singleton(){

}

public static getInstance(){

if(null == singleton){

synchronized(Singleton.class){

if(null == singleton){

singleton = new Singleton();

}

}

}

return singleton;

}

}

## 冒泡排序：

for（int i=0;i<sort.length-1;i++）{

for（int j=0;j<sort.length-1-i;j++）{

If（sort[j]<sort[j+1]）{

temp=sort[j];

sort[j]=sort[j+1];

sort[j+1]=temp;

}

}

}

## Sql语句：

分页查询每页3条查看第4页的数据

select s.\* from

(select t.\* , rownum rn from

(select \* from t\_score )t

where rownum<=12)s

where rn>=10

删除重复的记录

delete from t\_log tl where rowid <(

select max(rowid) from t\_log tg where tl.logid=tg.logid and tl.logname=tg.logname and

tl.logdate=tg.logdate and tl.logcount=tg.logcount and tl.typeid=tg.typeid

)

## 二叉树

## JDK版本特性！

Jdk1.7新特性?

1， Switch可以支持字符串,

2，泛型自动推断，

3，俩个char之间可以使用equals方法?

4，安全的加减乘除

5，map支持花括号传值

Jdk1.8新特性

1，允许接口中有默认方法实现??

2，函数式接口??

3，接口里允许使用方法体

4，允许：：调用方法???

5，加入全新的日期API（这些功能都放在java.time包下）提出了时间的加减乘除?

6，允许在类和接口上写注解

JDK1.9新特性

1，简化进程API??

2，轻量级的JSon?API??（Java.util包）

3，钱或货币的API（钱或货币之间的转换）?

4，改善锁争用机制（通讯服务器开房了海量的进程，链接客户端申请同一个资源）

## 接口与抽象类的区别？

1. 抽象类可以有构造方法，接口中不能有构造方法。

2. 抽象类中可以有普通成员变量，接口中没有普通成员变量

3. 抽象类中可以包含非抽象的普通方法，接口中的所有方法必须都是抽象的，不能有非抽象的普通方法。

4. 抽象类中的抽象方法的访问类型可以是public，protected和默认类型，但接口中的抽象方法只能是public类型的，并且默认即为public abstract类型。

5. 抽象类中可以包含静态方法，接口中不能包含静态方法

6. 抽象类和接口中都可以包含静态成员变量，抽象类中的静态成员变量的访问类型可以任意，但接口中定义的变量只能是public static final类型，并且默认即为public static final类型。

7. 一个类可以实现多个接口，但只能继承一个抽象类。

## get和post请求的区别？

答：

①get请求用来从服务器上获得资源，而post是用来向服务器提交数据；

②get将表单中数据按照name=value的形式，添加到action 所指向的URL 后面，并且两者使用"?"连接，而各个变量之间使用"&"连接；post是将表单中的数据放在HTTP协议的请求头或消息体中，传递到action所指向URL；

③get传输的数据要受到URL长度限制（1024字节）；而post可以传输大量的数据，上传文件通常要使用post方式；

④使用get时参数会显示在地址栏上，如果这些数据不是敏感数据，那么可以使用get；对于敏感数据还是应用使用post；

⑤get使用MIME类型application/x-www-form-urlencoded的URL编码（也叫百分号编码）文本的格式传递参数，保证被传送的参数由遵循规范的文本组成，例如一个空格的编码是"%20"。

## &和&&的区别。

&和&&都可以用作逻辑与的运算符，&&为短路与，&不是短路与。

另外&可以做为整数的位运算符

例1：对于if(str != null&& !str.equals(“”))表达式，当str为null时，后面的表达式不会执行，所以不会出现NullPointerException如果将&&改为&，则会抛出NullPointerException异常。

例2：If(x==33 &++y>0) y会增长，if(x==33 && ++y>0)不会增长

备注：这道题先说两者的共同点，再说出&&和&的特殊之处，并列举一些经典的例子来表明自己理解透彻深入、实际经验丰富。

## "=="和equals方法的区别？

他们的区别主要存在在引用数据类型上

==为比较两侧的对象是否同一对象，是用内存地址来比较的

equals是方法，默认是用内存地址比较，重写后，主要是用来比较两侧的对象的值是否相同，和equals方法中的实现有关

==可以两侧都为null，但equals左侧的引用指向的对象不能空，不然有NullPointerException

除非需要比较两个引用指向的对象是同一对象，一般都使用equals方法进行比较。尤其是String之类的值对象，另外，常量尽量放在比较的左侧

## Integer与int的区别

int是java提供的8种原始数据类型之一，意思整型，占用4字节。

Integer是java为int提供的封装类，是引用数据类型。

int的默认值为0，而Integer的默认值为null，即Integer可以区分出未赋值和值为0的区别，int则无法表达出未赋值的情况。

例如，要想表达出没有参加考试和考试成绩为0的区别，则只能使用Integer

在JSP开发中，Integer的默认为null，所以用el表达式在文本框中显示时，值为空白字符串，而int默认的默认值为0，所以用el表达式在文本框中显示时，结果为0，所以，int不适合作为web层的表单数据的类型。

在Hibernate中，如果将OID定义为Integer类型，那么Hibernate就可以根据其值是否为null而判断一个对象是否是临时的，如果将OID定义为了int类型，还需要在hbm映射文件中设置其unsaved-value属性为0。

另外，Integer提供了多个与整数相关的操作方法，例如，将一个字符串转换成整数，Integer中还定义了表示整数的最大值和最小值的常量。

## 面向对象的特征有哪些方面

1. 封装，隐藏内部实现，只暴露公共行为

2. 继承，提高代码的重用性

3. 多态，体现现实生活中相似对象的差异性

4. 抽象，抽取现实世界中相似对象的共同点

## linux中的命令：

1 ctrl + alt 切换系统

2 开机 power on 关机 power off

3 命令 ： 跳目录 cd / 根目录 cd.. 上一级目录

4 ls 插卡目录 mkdir 建文件夹 touch 建文件 cat 看文件内容 vi 编辑文件（insert模式，按esc 键退出 ：wq 保存并退出）

5 rm -rf 删除文件 su root 切换用户

ps -ef | grep :查看进程信息

vi:文件编辑命令

more:分页查看命令

top:常用的性能分析工具,能够实时显示系统中各个进程的资源占用状况

ifconfig:显示或配置网络设备的命令

ping:它通常用来测试与目标主机的连通性

rsync、scp：文件同步命令

## Jvm相关

栈溢出，就是当递归调用的时候没有临界退出条件

批量导入大量数据或者dom4j解析大的xml文件的时候

会出堆溢出，通过分段批量提交以及sax代替dom4j

java虚拟机有一个堆，堆是运行时数据区域所有的类和数组的内存都是从堆里分配的

堆是在java虚拟机启动的时候创建的，两种内存：堆和栈，堆是java开发人员使用的

栈是jvm自己使用的，内存结构分为：堆（逻辑上连续的，物理上不是连续的，存的是

类实例，和数组），虚拟机栈（基本的数据类型，引用对象），本地方法栈（jvm栈是为

java方法的执行提供服务，而本地方法栈是为虚拟机为本地方法提供服务的），方法区（持久代）

（加载了类的基本信息，常量池的属于方法区的一部分，string就是存在常量池中），

计算器（就是来选取要执行下一条字节码的指令）

gc垃圾回收机制，动态分配内存的大小，依靠垃圾回收机制来完成对分配内存空间的回收，

从而避免了内存溢出的问题，降低了程序员工作的复杂度，采用了分代回收算法，jvm将内存分为

年轻代和年老代，年轻代就是对象刚建立不久，生命周期也很短，而年老代对象创建的比较久了

而且生命周期也很长，对于年轻代 频繁 采用复制算法 而年老代比较少采用tracing算法

jvm的内存大小和操作系统有关，一般来说是32位处理器内存为2个g，64为的处理器就不会有有限制了

操作系统会给一个限制一般是2GB-3GB（一般来说Windows系统下为1.5G-2G，Linux系统 下为2G-3G）

java的监视和管理控制台：是我们在运行的时候监视各种jvm资源统计信息，使用与检测死锁，内存遗漏，可以连接到一个本地或远程的jvm可以用来监视线程的状态，内存的使用情况，垃圾收集等等等

jvm的调优：开启server模式，增堆的大小，以及持久代的大小，提高程序的运行效率，

设置初堆大小和最大堆的大小都为一样的值，避免了堆增长带来额外的压力，持久代也是一样

jvm中类的生命周期：加载、连接、初始化，使用，卸载

连接（校验，准备，解析）

在服务器启动的时候报内存溢出是因为方法区太小，也就相当于持久代的内存太小。

通过-XX:PermSize以及-XX:MaxPermSize来指定其大小

，可以解决这个问题。

引导类加载器：rt.Jar包

扩展类加载器：jre目录有一个ext下面的lib目录

系统类加载器：classPath的里面的东西

自定义类加载器:加载自定义类

## Jvm优化：

JVM 优化主要是解决java的 GC （垃圾回收）问题。

JVM 的使用过程中各代有，年轻带主要存放，新创建对象。 年老代，年老代存放从年轻代存活的 对象。Perm（持久代）用 于存放静态文件，如今Java类、方法等。一般持久代可以设置大一点。

GC优化的目的有两个：

1、将转移到老年代的对象数量降低到最小；

2、减少full GC的执行时间；

为了达到上面的目的，一般地，你需要做的事情有：

1、减少使用全局变量和大对象；

2、调整新生代的大小到最合适；

3、设置老年代的大小为最合适；

4、选择合适的GC收集器；

【垃圾回收（GC收集器）:串行收集器、并行收集器、并发收集器。

o 串行处理器：

--适用情况：数据量比较小（100M左右）；单处理器下并且对响应时间无要求的应用。

--缺点：只能用于小型应用

o 并行处理器：

--适用情况：“对吞吐量有高要求”，多CPU、对应用响应时间无要求的中、大型应用。举例：后台处理、科学计算。（例如 ERP 银行系统）

--缺点：应用响应时间可能较长

o 并发处理器：

--适用情况：“对响应时间有高要求”，多CPU、对应用响应时间有较高要求的中、大型应用。举例：Web服务器/应用服务器、电信交换、集成开发环境。（例如互联网网站）】

5、设置jvm堆大小 ，32bit 1.5-2G ，64bit 可以超过 2G ，新版的JDK 每个线程的堆大小在1M改变这个线程所占用的堆大小，可以生成更多的线程，一般项目里线程数不能超过5000个。

## 多线程介绍、列举线程池和业务场景：

多线程基本介绍：http://www.cnblogs.com/yxt9322yxt/p/4804026.html

1）.什么是多线程

1个进程中可以开启多条线程，每条线程可以并行（同时）执行不同的任务

进程 ->车间，线程->车间工人

多线程技术可以提高程序的执行效率

比如同时开启3条线程分别下载3个文件

2）.多线程的原理

同一时间，CPU只能处理1条线程，只有1条线程在工作（执行）

多线程并发（同时）执行，其实是CPU快速地在多条线程之间调度（切换）

如果CPU调度线程的时间足够快，就造成了多线程并发执行的假象

思考：如果线程非常非常多，会发生什么情况？

CPU会在N多线程之间调度，CPU会累死，消耗大量的CPU资源

每条线程被调度执行的频次会降低（线程的执行效率降低）

3）线程池：

是指在初始化一个多线程应用程序过程中创建一个线程集合，然后在需要执行新的任务时重用这些线程而不是新建一个线程。线程池中线程的数量通常完全取决于可用内存数量和应用程序的需求。然而，增加可用线程数量是可能的。线程池中的每个线程都有被分配一个任务，一旦任务已经完成了，线程回到池子中并等待下一次分配任务。

2）Java提供的四种线程池的好处在于：

a. 重用存在的线程，减少对象创建、消亡的开销，性能佳。

b. 可有效控制最大并发线程数，提高系统资源的使用率，同时避免过多资源竞争，避免堵塞。

c. 提供定时执行、定期执行、单线程、并发数控制等功能

3）Java通过Executors提供四种线程池，分别为：

newCachedThreadPool

创建一个可缓存线程池，如果线程池长度超过处理需要，可灵活回收空闲线程，若无可回收，则新建线程。

newFixedThreadPool

创建一个定长线程池，可控制线程最大并发数，超出的线程会在队列中等待。

newScheduledThreadPool

创建一个定长线程池，支持定时及周期性任务执行。

newSingleThreadExecutor

创建一个单线程化的线程池，它只会用唯一的工作线程来执行任务，保证所有任务按照指定顺序(FIFO, LIFO, 优先级)执行。

4）多线程的使用场景：http://blog.csdn.net/hll814/article/details/50816268

1、 常见的浏览器、Web服务(现在写的web是中间件帮你完成了线程的控制)，web处理请求，各种专用服务器(如游戏服务器)

2、 servlet多线程

3、 FTP下载，多线程操作文件

4、 数据库用到的多线程

5、 分布式计算

6、 tomcat，tomcat内部采用多线程，上百个客户端访问同一个WEB应用，tomcat接入后就是把后续的处理扔给一个新的线程来处理，这个新的线程最后调用我们的servlet程序，比如doGet或者dpPost方法

7、 后台任务：如定时向大量(100W以上)的用户发送邮件；定期更新配置文件、任务调度(如quartz)，一些监控用于定期信息采集

8、 自动作业处理：比如定期备份日志、定期备份数据库

9、 异步处理：如发微博、记录日志

10、页面异步处理：比如大批量数据的核对工作(有10万个手机号码，核对哪些是已有用户)

11、数据库的数据分析(待分析的数据太多)，数据迁移

12、多步骤的任务处理，可根据步骤特征选用不同个数和特征的线程来协作处理，多任务的分割，由一个主线程分割给多个线程完成

13、desktop应用开发，一个费时的计算开个线程，前台加个进度条显示

14、swing编程

## jQuery的10个常用方法,常用选择器(五大类),ajax常用的属性

常用方法:

1,$("#ID").val(); //取value值

2,$("#ID").val("xxx"); //赋值

3,$("#ID").text(); //相当于取innerText

4,$("#ID").text(""); //相当于赋值给innerText

5,$("#ID").html(); //相当于取innerHTML

6,$("#ID").html(""); //相当于赋值给innerHTML

7,$("#ID").add();//将元素添加到匹配元素的集合中

8,$("#ID").hide(); //隐藏

9,$("#ID").show(); //显示

10,$("#ID").attr(key,value);//取得或设置匹配元素的属性值

常用选择器:

1,id选择器$("#myELement")选择id值等于myElement的元素，id值不能重复在文档中只能有一个id值是myElement所以得到的是唯一的元素

2,标签选择器$("div")选择所有的div标签元素，返回div元素数组

3.类选择器$(".myClass")选择使用myClass类的css的所有元素

4.通配符选择器$("\*")选择文档中的所有的元素

5.属性选择器$("[href='#']") 选取所有带有 href 值等于 "#" 的元素。

Ajax的属性:

1.url:?要求为String类型的参数，（默认为当前页地址）发送请求的地址。

2.type:?要求为String类型的参数，请求方式（post或get）默认为get。注意其他http请求方法，例如put和delete也可以使用，但仅部分浏览器支持。

3.async:为Boolean类型的参数，默认设置为true，所有请求均为异步请求。如果需要发送同步请求，请将此选项设置为false。注意，同步请求将锁住浏览器，用户其他操作必须等待请求完成才可以执行。

4.data:前台往后台传送的数据信息,可以是form表单或者单个数据信息

5.dataType:json或者text,规定请求后台返回前台时的数据类型

6.success：为Function类型的参数，请求成功后调用的回调函数

7.error:为Function类型的参数，请求失败时被调用的函数

## Tomcat优化

1.使用64位的tomcat和jdk

2.开启server模式

3.通过-Xms和-Xmx设置初始堆大小和最大堆大小，通常将两个值设置为一样，避免堆空间不断增大和缩小所带来的性能损耗

4.启用gzip压缩

5.通过maxThreads增加tomcat的最大并发数，将其设置为500

## Hibernate优化：

在处理大数据量时，会有大量的数据缓冲保存在Session的一级缓存中，这缓存大太时会严重显示性能，所以在使用Hibernate处理大数 据量的，可以使用session. clear()或者session. evict(Object) ,在处理过程中，清除全部的缓存或者清除某个对象。

通过使用Hibernate的一级缓存，二级缓存，查询缓存等来提高性能

Hibernate可以通过设置hibernate.jdbc.fetch\_size，hibernate.jdbc.batch\_size等属性，对Hibernate进行优化。

Batch Size是设定对数据库进行批量删除，批量更新和批量插入的时候的批次大小，Batch Size越大和数据库交互的次数就越少，速度就越快，

但也不要无限的大下去，通常选择一个合适的值，如100条；其次我们在进行大批量数据的导入操作时，可以结合batchsize进行分段批量提交，从而达到最优效果。

## 序列化和反序列化

序列化：序列化就是将内存中的对象持久化到本地文件的过程

为什么要序列化：通常在http网络传输过程中，是不支持对象传递的，所以通过序列化，将对象转换成可以被传输的流，方便对象信息传递

如何将一个对象序列化：1、类要实现 Serializable接口；2、通过ObjectOutputStream的writeObject()将对象写到文件中

反序列化：将文件中存储的对象信息读取后写到内存中

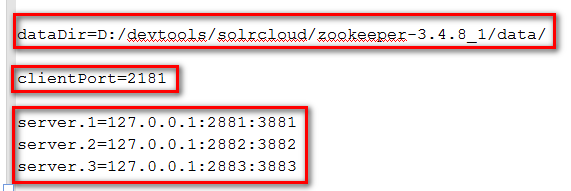
## 集群

Zookeeper集群

1、分别配置3个解压好的zookeeper，在每个zookeeper文件夹下创建一个data目录，在data文件夹下创建一个文件（myid），文件的内容就是此zookeeper的编号（1、2、3）

2、分别把3个zookeeper下conf目录下的zoo\_sample.cfg文件复制一份改名为zoo.cfg

3、分别修改3个zookeeper下的zoo.cfg的配置



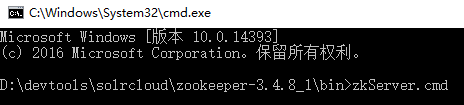
dataDir：分别对应3个zookeeper各自新建的data路径

clientPort：当前每个zookeeper的端口号，要保证唯一性

server.(1,2,3)：3个zookeeper各自的ip地址：通讯端口：选举端口

4、分别启动3个zookeeper。使用dos进入每个zookeeper/bin目录下。

启动方式如下，必须这么启动，双击zkServer.cmd启动无效



到此，zookeeper集群配置完成

Solr集群

1、创建4个tomcat实例，分别修改其端口。8981-8984

2、按照之前的流程把solr.war分别部署到4个tomcat/webapps下

                  注意解压开solr.war，然后删除solr.war源包

            注意把jar包拷贝过来（example/lib/ext/\*.jar）

3、分别创建4个solrhome(1,2,3,4)，把solr源包下的example\solr\\*拷贝到每一个solrhome下

4、分别修改4个tomcat下solr的web.xml，分别指定其对应的那个solrhome的位置。

5、把任一个solrhome中的配置文件上传到zookeeper集群。使用zookeeper的客户端上传

6、分别修改solrhome(1,2,3,4)下的solr.xml文件，指定当前实例运行的ip地址及端口号。

7、分别修改每一台solr的tomcat 的 bin目录下catalina.bat文件，向文件中加入-DzkHost指定zookeeper集群服务器地址

8、分别启动4台tomcat，访问任意一台tomcat

9、创建一个两片的collection，重新访问，删除collection1，重新访问

 Tomcat集群

1、复制两个tomcat，在tomcat中将webapps中的ROOT文件夹都删掉（注意：使用在server.xml文件中配置<Context path="" docBase="e:/ssh-login"/>，启动tomcat时，将在webapps文件夹中生成ROOT文件夹，访问时不加项目名）

2、修改server.xml文件中的端口号，第一个tomcat中修改端口号为8081

3、启动tomcat

4、修改nginx-1.9.14文件夹下的conf文件夹下的nginx.conf文件（http标签、本地服务器连接端口、请求拦截）

Mysql主从：

1、首先要安装两台MySQL服务；端口号不能一致。配置主库mastar，server-id = 1

2、指定要同步的数据库，给主库添加一个用户，并指定replication权限，在主数据库里面运行show master status;记下file和position字段对应的参数。

3、配置从库，指定同步的数据库，首先要先执行stop slave;命令，先停止从库的复制功能；然后设置主从库的一些参数，要对应主数据库里查出来的file和position两个参数，最后开启从库复制功能，start slave；

nginx热备：

使用nginx+keepalived双机热备的方式来实现负载均衡调度器的高可用性，其中keepalived主要是解决了nginx的单点故障，用来检测服务器状态，如果一台web服务器挂了，keepalived将检测到，并将有故障的机器剔除掉，然后人工进行修复。

# 非专业知识：

## 大学情况：

学校名称：西安外事学院（外事）

学校地址：有南北校区，都在鱼化寨

出生日期：1992年11月16日

上小学日期：1998年9月 6岁 六年

上初中日期：2003年9月 12岁 三年

上高中日期：2006年9月 15岁 三年

上大学日期：2009年9月 18岁 四年

毕业时间：2013年7月 22岁

入学时间：2009年9月

毕业时间：2013年7月

校长名字：陈爱民

学院：工学院

专业名称：计算机信息与技术

专业课: 数据结构、计算机组成原理、操作系统、软件工程、计算机程序设计、计算机网络、计算机图形学、多媒体技术、数据库原理、面向对象技术、图象处理技术、电路原理、模拟电子技术、数字逻辑技术、计算方法、编译原理等。

院系名称：计算机学院

学历证书：本科

学位:学士学位

在校学生有：12000余人

有教师：1000左右人

学校的院系有：商学院、文学院、医学院、工学院、影视艺术学院等10几个学院，

本院专业：计算机科学与技术、物联网工程、电气工程及其自动化、电子信息工程、交通运输、产品设计六个本科专业

路线:40路，外事北校区站下车

医学院护理专业；商学院人资、会计；文学院学前教育

三、主要课程

主干课程：高级语言程序设计、UI设计（基础/高级）、Linux操作系统、数据结构与算法、计算机网络原理、软件工程、数据库原理与应用、SQLite数据库技术、Java（基础/高级）、Web网页技术、Java Web开发技术、Web框架技术、Android开发（基础/高级）、软件分析与测试

主要实践、创新环节：社会调查、Linux操作系统实训、高级语言程序设计实训、数据库实训、Web网页技术实训、专业综合实训、毕业论文（设计）

主要专业实验：Linux操作系统实验、高级语言程序设计实验、数据库实验、Java程序设计实验、静态/动态网页设计实验，UI交互设计实验、Android开发实验、Java Web开发实验、Web框架实验等

## 上家公司：

公司简介：北京思前软件有限公司（简称就好）

公司地址：北京海淀区上地七街1号汇众大厦2号楼106室

住址到公司地址：乘坐592路公交，从育新地铁站上车——上地七街站下车 时间大概40分钟左右，打车大概20左右

公司规模：20-99人,我们项目组9 一个项目经理，5名开发人员，2名前台，一个运维

入职时间：2013-8

## 有没有上过保险：

上个公司说是可以要折现。我就没入

## 没有上五险一金的原因：

没有在北京买房的目的，交了五险一金还不如每个月多发一些现金比较来的实际

## 五险一金的名称：

养老保险，医疗保险，失业保险，工伤保险，生育险，住房公积金

## 三个词形容自己：

认真细致，团队协作能力强，抗压能力强

## 谈谈你的对软件行业发展前景的理解：

现在的生活越来越离不开网络和便捷的生活方式，软件的行业就是顺应了这一特点，通过开发的软件简化了人们一些生活复杂度，所以在不久的将来必将成为科技的前端和人们生活的密切相关。

## 为什么离职：

公司最近转型，转向.net/php了，java这方面的项目也少了，所以想换个环境，不断提高充实自己。我觉得咱们公司有自己的企业文化和实力（具体到时候自己网上查），能够给我提供一个展现自己能力的价值的平台，所以我给咱们公司投递了简历。

## 如何看待你上家公司：

每个公司都有自己的一个企业文化，从上一家公司我有深刻的体会，就是遇事要向内归因，外界的给我们带来的压力只能是客观因素，从本身去找解决办法才是我们提高的一个重要标志，逐步让我成为一个有责任心，有目标的人。和之前的同事的交流也让我学到非常多的知识和扩展交际圈，明白更多的沟通方式。

## 你凭什么要这么高的薪资?

首先我既然敢要这么多钱，那么我自信我有这样的实力能给公司带来同等甚至更大的经济效益，我正值壮年，精力旺盛，能够在高压力下正常的工作，曾在项目上线时连续两天通宵直到项目正常运行，在工作时，项目经理让我研究东西，我能在较快的时间里学会并熟练应用，把这项技术教给同事，让他们快速学会这项技术

## 四年涨薪：

2013年实习的时候是6500，税后大概6000左右，转正7500左右

到2015上半年涨了5000左右 ，总12000左右

2016跳槽一次涨到16000 ，税后大概15000，到今年涨到17000

## 跳槽的看法:

不能频繁跳槽，适当的跳槽是可以理解的，为了个人的一些东西。

## 你的五年规划

如果我被贵公司录用的话,肯定先快速的适应公司新环境.与同事处好人际关系，

然后在完成公司给的任务之后学一些新的技术，来提升自己的能力。如果贵公司晋升机制好的话，我会考虑向项目经理的方面去发展。

## 谈谈你对加班的看法：

加班其实是每个公司遇到的问题，之前我也提过我是一个能够以大局为重的人，为了公司的利益和项目组的进度我完全可以加班来完成，但是在工作中我会规划好自己的进度按时完成任务，即使在我的任务完成的前提下，我也会和项目组待在一起帮助他们完成项目任务。

## 分库分表：

为什么分表：一张表中的数据太多了、为什么分库：一个数据库中的表太多了、如果不分库分表的后果：检索速度慢、层次不清晰。

在这工作几年的过程中让我明白到,当遇到困难时不去退缩勇于面对才能让自己更快的成长，学会向内归因，不推卸责任和错误，积极主动承担责任和承认错误，吸取别人的优点来补足自己的缺点和不足，避免下次犯同样的错误。闲暇时喜欢上一些技术类的网站：比如开源中国、CSDN，推酷、博客园等网站，关注一些前沿的技术。因为我相信，学无止境，不进则退。

## 最后你还想有什么了解吗？

* **咱们公司的五险一金是怎么交的？**
* **咱们公司开发的主力框架什么？**
* **咱们公司的晋升制度是什么？**
* **咱们公司用的什么技术框架/开发工具/数据库？**
* **咱们公司在开发什么类型的项目？**