# 第二简历

## 学校有关

河北传媒学院

从石家庄运河桥客运站沿东街北站口上车赵卜口站导车做520路到河北传媒学院

地址：河北省石家庄市栾城区兴安大街109号

附近：苏东坡祖籍纪念馆、栾城-天山国家农业公园

校训：传媒是社会的良知，人类的道义

院歌：《满天星》

院徽：



含义：

以“人”字为造型基础，紧扣校训——“成人成才”；两侧图形对称，代表一真一实，寓意永远“求真求实”；整体造型似一只展翅欲飞雄鹰，自尊、自信、傲视天下；中国古代“回形纹”，寓意教育根植于博大精深的中国传统文化；插上“翅膀”的人，代表展翅的精英人，腾飞的精英事业；阶梯上升感的造型，预示教育循序渐进、不断攀升；线性结构纵横通畅，寓意河传四通八达，多元化发展；远观整体图形又似“古钟”造型，暗喻不鸣则已、一鸣惊人；外观又似金榜题名的“状元帽”。

计算机科学与技术

学制：四年。

学生毕业授予工学学士学位。

专业课程：

电路基础、数字电路、计算机组成原理、汇编语言程序设计、计算机网络、C（C++）语言程序设计、数据结构、操作系统、数据库原理、单片机原理及应用、WEB数据库、JAVA、软件工程、计算方法、微机原理及应用等。

## 公司有关

公司董事：[王晨曦](https://www.tianyancha.com/human/2072166768-c24796391" \t "https://www.tianyancha.com/company/_blank)

**经营范围：**技术服务、技术转让、技术开发；计算机系统服务；经济信息咨询；投资咨询；劳务服务（外包）

公司行业：北京湛腾世纪科技有限公司成立于2011年12月，是一家全方位IT服务及行业解决方案提供商。

公司名称：**北京湛腾世纪科技有限公司**

地址：北京市丰台区南三环西路16号2号楼13层1606

路线：昌平线上车-13线-4号线马家堡下车

## 项目有关

你好，我叫李德， 来自河北石家庄，毕业于河北传媒学院计算机科学与技术专业，从2015年工作毕业之前在北京湛腾世纪科技有限公司实习，刚开始也就是做一些简单的代码编写，毕了业之后也就一直呆在湛腾世纪科技有限公司做java开发上班，我在公司常用的框架有SSH、SSM一些主流框架。

公司常年对外承包项目，最近做的项目是合家欢vip购物平台这个网站是帮合家欢连锁企业定制的一个vip客户购物平台。该平台，仅面向VIP会员服务，为VIP会员提供方便快捷专一的奢侈品购物平台，以及网上私人服务订制。整个网站的设计，采用的是分布式架构设计，其中构成整个网站的各个子模块包括了后台管理、首页、用户、搜索、商品详情、购物车、订单、支付、会员、快速下单等模块。

在软件架构上我们使用的是soa架构（SOA是Service-Oriented Architecture的首字母简称，它是一种支持面向服务的架构样式。从服务、基于服务开发和服务的结果来看，面向服务是一种思考方式。其实SOA架构更多应用于互联网项目开发。）

）实现了控制层和服务层的分离，后端采用的框架是ssm+dubbox，前端采用的是AngularJs+BootStrap,使用了springSecurity控制了用户的登录和权限.

我主要负责：用户模块的开发、首页模块、商品的管理模块、商品的详情、商品的购物车和订单。

### Dubbox原理

#### Bubbox是什么？

Dubbo是一个分布式服务框架，致力于提供高性能和透明化的RPC远程服务调用方案，以及SOA服务治理方案。简单的说，dubbo就是个服务框架，如果没有分布式的需求，其实是不需要用的，只有在分布式的时候，才有dubbo这样的分布式服务框架的需求，并且本质上是个服务调用的东东，说白了就是个远程服务调用的分布式框架（告别Web Service模式中的WSdl，以服务者与消费者的方式在dubbo上注册）

#### Dubbox能做什么？

1.透明化的远程方法调用，就像调用本地方法一样调用远程方法，只需简单配置，没有任何API侵入。

2.软负载均衡及容错机制，可在内网替代F5等硬件负载均衡器，降低成本，减少单点。

3. 服务自动注册与发现，不再需要写死服务提供方地址，注册中心基于接口名查询服务提供者的IP地址，并且能够平滑添加或删除服务提供者。

Dubbo采用全Spring配置方式，透明化接入应用，对应用没有任何API侵入，只需用Spring加载Dubbo的配置即可，Dubbo基于Spring的Schema扩展进行加载。

Dubbox实现了分布式服务的提供者和消费者。通过zookeeper的注册中心，

实现提供者和消费者的映射。消费者调用提供者提供的服务。

### AngularJs四大特征

angularjs四大特性：

　　1、MVC模式：

　　　　Model:数据,其实就是angular变量($scope.XX,$rootScope.XX);

　　　　View:数据的呈现,Html+Directive(指令);

　　　　Controller:操作数据,就是function,数据的增删改查;

　　2、双向绑定:

　　　　方向一:Model--->View

　　　　　　　　{{Model数据}} 或<XXX ng-xxx="Model数据">  Model变View跟着变;

　　　　方向二:View--->Model

　　　　　　　　<表单控件 ng-model="Model数据名">   View变Model跟着变;

　　3、依赖注入:

　　　　依赖注入(Dependency Injection,简称DI)是一种设计模式, 指某个对象依赖的其他对象无需手工创建，只需要“吼一嗓子”，则此对象在创建时，其依赖的对象由框架来自动创建并注入进来,其实就是最少知识法则;模块中所有的service和provider两类对象，都可以根据形参名称实现DI.

　　4、模块化设计:

　　　　高内聚低耦合法则,

　　　　1)官方提供的模块           ng、ngRoute、ngAnimate、ngTouch

        　2)用户自定义的模块        angular.module('模块名',[ ])

#### AngularJS在Web中实现原理

我们在前端使用AngularJS框架，主要使用了AngularJS的MVC模式和模块化的思想，实现了前端JS代码分层、分模块，形成了和后端一样的，服务层和控制层。然后使用依赖注入把Controller层需要的服务注入进去。最后，使用双向数据绑定来异步加载数据。

## 用户模块

### SpringSecurity安全登录

Spring Security 简介   
Spring 是一个非常流行和成功的 Java 应用开发框架。Spring Security 基于 Spring 框架，提供了一套 Web 应用安全性的完整解决方案。一般来说，Web 应用的安全性包括用户认证（Authentication）和用户授权（Authorization）两个部分。用户认证指的是验证某个用户是否为系统中的合法主体，也就是说用户能否访问该系统。用户认证一般要求用户提供用户名和密码。系统通过校验用户名和密码来完成认证过程。用户授权指的是验证某个用户是否有权限执行某个操作。在一个系统中，不同用户所具有的权限是不同的。比如对一个文件来说，有的用户只能进行读取，而有的用户可以进行修改。一般来说，系统会为不同的用户分配不同的角色，而每个角色则对应一系列的权限。

#### 运营商登陆与安全控制功能

我们实现登录使用spring Security，我们都是使用spring-security.xml来配置的。

里面包含了登录的账户信息和角色 、登录页面地址、登录成功跳转地址、登录失败跳转地址。

如果要页面能够加载框架页，需要设置框架页的策略为SAMEORIGIN

为了让我们登录之后，访问是默认的页面，在登录页面中，action=”/login” method=”post”, 用户名是username，密码是password都是固定的。

#### 获取登录用户名功能

页面加载的时候，去访问后台，后台使用SecurityContextHolder的方法能够从SpringSecurity中获得登录的用户名，返回给前端，使用angularJS的表达式显示。

#### 商家系统登录与安全控制

我们要用数据库数据来实现登录，首先我们需要创建一个实现了UserDetailsService接口的实现类，并复写loadUserByUsername。在该方法中，首先获取登录的用户名，然后根据用户名去查询用户的密码，最后，把用户名和密码交给SpringSecurity与我们输入的用户名和密码进行匹配。如果匹配成功，跳转到login-page指定的页面。

#### 实现密码加密

我们使用BCrypt算法实现密码加密。当我们添加用户信息的时候，创建BCrypt密码加密的对象，然后调用加密encode方法，实现加密，放到数据库。最后，还需要在springsecurity的配置文件中指定加密算法。

## 商品管理模块

#### 商品的三级分类显示

商品三级分类的显示，我们会根据分类的父id（parent\_id）查询分类。

默认情况，分类页面显示顶级分类，也就是parent\_id=0的分类。当我们

点击某个顶级分类的时候，会根据这个顶级分类的id，去查询这个顶级分类

的二级分类。当我们点击某个二级分类，根据这个二级分类的id，去查询它的三级分类。

首先在后台定义一个根据parent\_ id查询分类列表的方法，然后在前端的js页面的controller层，接收传递过来的parentId，调用后台的方法。

当进入三级分类页面的时候，默认显示顶级列表，使用ng-init在页面初始化的时候，调用js的controller层，通过service获得后台查询到的parent\_id=0的数据的集合。如果成功获得就将数据添加到$scope的变量中，使用ng-repeat循环集合，最终使用angularjs的表达式显示。

二级分裂列表显示，通过获得点击分类的id，这个id就是二级分类的parentid，同样道理，查询二级分类内容。三级依然。

#### 面包屑导航

根据商品的三级分类，面包屑导航也分为三级。在进入分类页面的时候，页面中

默认显示顶级分类，面包屑导航显示固定字符--“顶级分类”。当我们查看二级分类的时候，

面包屑导航显示我们选中的顶级分类的名称。当我们查看三级分类的时候，面包屑导航显示的是我们选中的二级分类的名称。

实现面包屑导航三级分类，首先定义一个分级的级别，默认是1。当我们点击”查看下级分类”的按钮时候，能够获取到当前分类对象，并把当前分类对象的名称显示在面包屑导航中。

当我们点击某个面包屑导航，首先设置分类级别，然后根据不同的分类级别，给不同的分类对象设置内容。

使用ng-if判断当前的分类级别，如果当前的级别不等于3的时候才显示“查询下级”按钮。

#### 商品的录入

在商家后台实现商品录入分为：基本功能、商品介绍、商品的图片上传、选择商品分类、品牌选择、扩展属性、规格选择、SKU商品信息、是否启用规格。

基本功能：包括商品名称、副标题、价格、包装列表、售后服务

商品录入我们要同时操作两张表，tb\_goods商品表和tb\_goods\_desc商品描述表。

首先创建一个包装类，里面包含两个属性goods商品的基本信息，goodsDesc商品的扩展信息。

当我们输入内容，点击保存的时候，将商品信息和商品的描述信息一起添加包装类中。我们将商品的名称，价格，副标题等一些基本信息插入到tb\_goods商品表中，并通过返回插入数据的id，这个id是通过mybatis的sql映射文件，里面insert标签中包含selectKey标签调用LAST\_INSERT\_ID的方法获得的。获得这个id之后，以这个id为外键向tb\_goods\_desc表中插入商品的描述信息。这里需要注意的是，商品介绍的内容是通过富文本编辑器获得的。要使用富文本编辑器，在页面中导入富文本编辑器的css和js，然后在页面中添加js的代码，初始化富文本编辑器。然后在页面中通过js的edit.Html()用来获取编辑器的内容。

商品图片上传

我们的图片是通过专门的文件服务器FastDFS来管理的。FastDFS是可以放置图片和文件的服务器，我们录入的所有的图片信息全部放在这个文件服务器上。图片上传的第一步，我们要在工程中添加文件上传的依赖fastdfs和fileupload。并添加工具类FastDFSClient。然后在商家后台添加配置文件application.properties（服务器地址），fdfs\_client.conf(图片上传路径)。并在springMVC的配置文件中配置图片上传解析器。

第二步、在前端的js页面，我们通过AngularJS实现上传，这里面需要特别设置两个属性：Content-Type header属性，这个属性的默认值是application/json。我们如果要上传图片，需要把这个属性设置为null，这样就会使用 multipart/form-data值。另外需要

设置transformRequest: angular.identity，这个属性主要是上传对象序列化。

当点击上传按钮的时候，我们在后端通过FastDFS的客户端上传图片，并获得上传地址。最后在添加商品描述对象的时候，把图片地址插入到tb\_goods\_desc表中的item\_images字段中。

在商品录入界面实现商品分类的选择（三级分类）

当用户选择一级分类后，二级分类列表要相应更新，当用户选择二级分类后，三级列表要相应更新。

#### 一级分类下拉选择框

页面启动时候，查询一级分类，也是parentId=0的分类，使用ng-options循环展示分类的id和名称。

#### 二级分类下拉选择框

当一级分类选中的时候，使用angularJS的内置对象$wathch获得选中的一级分类的id值，

以这个id值作为二级分类的parent\_id值，查询出二级分类。然后使用ng-options循环展示分类的id和名称。

#### 三级分类下拉选择框

当二级分类选中的时候，使用angularJS的内置对象$wathch获得选中的二级分类的id值，

以这个id值作为三级分类的parent\_id值，查询出三级分类。然后使用ng-options循环展示分类的id和名称。

#### 读取模板ID

当三级分类选中的时候，使用angularJS的内置对象$wathch获得选中的三级分类的id值，

以这个id值查询当前三级分类对应的模板id。然后使用angularjs的双括号表达式展示

#### 品牌选择

当模板id获得值之后，查询模板对象，获得模板对象中保存的品牌字段的json格式数据，

然后把json格式数据转化成js对象。然后使用ng-options循环展示品牌的id和名称。

#### 扩展属性

当模板id获得值之后，查询模板对象，获得模板对象中保存的扩展属性字段的json格式数据，然后把json格式数据转化成js对象。使用ng-repeat循环展示。

#### 显示规格及规格选型

[{"key":value,"array":[{"key":value},...]},...]

要实现前端效果，首先

需要考虑前端页面显示的数据结构。当数据结构是json的对象集合，其中的

对象属性又是对象集合。也就是集合对象中对象属性包含集合对象。这种形式，对应着Java的

List<Map>形式。

通过Angularjs的$watch的内置对象，监控模板id的变化。当我们在前台选中模板id的时候，

通过模板id找到规格id，然后通过规格的id找到对应的规格选项，然后把规格选项集合put到

规格map中，就拼装成了我们需要的对象属性包含集合对象形式。

最终，用ng-repeat进行循环展示。

#### 保存选中规格选项到变量中

因为规格选项保存在tb\_goods\_desc表的specification\_items字段中，这个字段的格式是[{“attributeName”:”规格名称”,”attributeValue”:[“规格选项1”,... ] } , ...]

这个格式json数组中包含对象集合，单个对象中又包含对象数组。

当我们点击某一个规格选项的时候，首先判断数组中是否这个规格对象，如果包含这个规格对象，我们就获取这个规格对象并添加新的规格选项。否则，如果不包含这个规格对象，我们就构造一个新的规格对象。

当我们取消选中规格选项的时候，首先判断这个规格选项是否是该规格对象中最后一个规格选项，如果是，就移除这个规格对象。如果不是，就从规格对象中移除这个规格选项。

#### SKU商品信息

要实现前端效果，首先需要考虑前端页面单条记录显示的数据结构。

[{spec:{},price:0,num:99999,status:'0',isDefault:'0' } ,...]

这个json格式数组中包含对象集合，单个对象中又包含对象。

根据刚才保存的规格选项，循环每个规格对象，里面包含规格名称和规格选项值。

循环SKU集合中的每个对象，通过js的深克隆，克隆新的SKU对象，把循环的每个规格名

和它下面的每一个规格选项循环添加到克隆的SKU对象中。最终，把克隆的每个SUK对象添加

到一个集合中。最后，对SKU集合对象循环展示。

当我们点击提交的时候，会获得每一个SKU对象，进行循环添加到tb\_item表中。其中，SUK的标题是

商品名称+规格名称

#### 启用规格

首先，设置启动规格的单选框的值，当选中的时候使用ng-true-value设置为1，未选中的时候使用ng-false-value设置为0。当我们启用规格的时候，使用ng-if来判断是单选框是否被选中，如果选中显示规格选项，否则隐藏。

当我们未启用规格，就保存单个SKU信息。当启用规格，才按照SKU的条数进行保存。

### 商家后台显示商品列表

通过SpringSecurity获得当前登录用户的sellerId，通过这个sellerId获取商品表里

这个用户的所有商品。

商品的状态分为未审核','已审核','审核未通过','关闭'。在js中定义一个数组，按照顺序

保存商品状态。根据商品的状态码获得数组中对应的商品状态。

首先从数据库中查询所有商品的分类名称，以id为键，以名称为值保存到数组中。

根据分类的id，从数组中找到对应的商品分类名，并显示。

### 商品的修改

商品修改，实质上是修改了商品表，商品描述表，商品SKU表。

在进行具体修改的时候，首先获得商品信息。获得商品信息分为四部分；

商品基本信息，上传的商品图片，商品扩展属性和商品规格选项四部分。

1》商品的基本信息默认直接显示，通过edit.html()获得商品的描述信息。

2》显示图片的商品列表，通过获取goodDesc表中ItemImages字段内容，

然后把图片的json数据转化成Js对象，然后显示。

3》同理，显示扩展属性，通过获得goodDesc表中**customAttributeItems字段内容，**

然后把扩展属性的json数据转化成Js对象，然后显示。

里面包含了一个错误，当我们显示扩展属性的时候，首先获得了扩展属性，

但是在监控模板id发生变化的时候，会把扩展属性清空。这时候，我们在监控模板id

的代码中，添加判断是否获得修改的id值。如果获得了id值，说明是修改，不要清空扩展属性。

4》同理，规格选项也是这么获得。在规格选项中，回显默认选中的值，首先获得被选中的规格选项的集合，

然后在集合中查找是否包含当前的规格选项，如果包含，就设置单选框的ng-check为true，表示单选框选中。

通过商品id，查询对应的SKU列表，在前端保存到SKU集合中，然后循环展示。

当完成修改商品之前，首先判断获得的id对应的商品属于当前登录用户，

且修改的商品属于当前用户，才能修改。

最后，把所有修改的商品表和商品描述的信息重新保存，其中商品记录状态设置为未审核状态。最后，删除SKU表中的原数据，重新保存新修改的SKU列表。

#### 商家后台商品删除

就是修改商品表中是否删除的isDelete字段，设置为1，代表商品已经删除。

设置查询操作已经删除的数据，排除在查询之外。

#### 商家后台商品上下架

对于审核过商品，有上架和下架两种状态。上架代表可以销售，下架代表不能销售。

修改商品表的is\_marketable字段。当设置1表示上架、0表示下架。

## 搜索模块

#### 搜索实现的原理

网络爬虫（网络蜘蛛）去各个网站上抓取网页，然后保存到服务器上，通过分词器,分析网页中的出现最多的关键词，按照关键词出现的次数进行排名。当我们搜索，实际上就是关键词出现的次数最多对应的网页的排名。

#### 配置Sorl环境

我们网站实现搜索，是使用了solr。Solr是一个实现搜索的开源服务。它本身是一个war包，可以放到tomcat中。

我们配置中文分词器IK Analyzer，首先添加分词器的jar包，然后把扩展字典、停用字典、分词器的配置IKAnalyzer.cfg.xml复制到WEB—INF/class目录下。最后，修改 Solrhome下的 schema.xml 文件，配置一个 FieldType，使用 IKAnalyzer。

我们根据搜索的业务需要，指定了业务域，比如标题，图片，价格，规格等。首先业务域的定义，常用属性有域的名字（索引库的字段名），域的类型type，是否索引index，是否储存store，是否必须require，是否多值。

定义域分为一般域<field name="item\_goodsid" type="long" indexed="true" stored="true"/>，

复制域（通过复制域可以实现搜索多个域中的关键字）

动态域（把类似域名的域，设置为一个独立的域）

#### 实现Sorl的数据导入

首先，去指定pojo类中属性和索引库域的对应关系。然后查询数据库中所有需要导入的数据。最终通过solrTemplate对象的saveBeans（）把数据导入到索引库中。

#### 使用SpringDataSorl实现简单查询

首先创建一个搜索对象Query，然后创建一个条件对象Criteria，设定查询的域以及查询的关键字。并添加查询对象中。最终调用sorlTemplate的queryForPage（）执行查询。

## 首页模板管理

#### 首页广告显示模块

我们实现首页的广告显示，因为首页访问量比较大，我们可以使用两种方式来减小服务器的压力。一种方式是使用Redis实现缓存，一种是实现页面的静态化。

我们使用redis缓存，将一些前台页面展示的数据放到redis中，减少访问数据库的次数，减缓服务器的压力。

我们首先在linux里面安装Redis、然后连接Redis。

使用Redis实现首页广告缓存:我们使用SpringDataRedis来操作Redis。当首页加载的时候，访问广告数据，首先去redis中查询是否有该类广告，如果有，直接从redis获得，不需要查询数据库。如果没有，就查询数据库，并把数据添加到redis中，下次查询时候，就可以从redis中获得。

更新缓存，当添加广告的时候，把广告所属的类的缓存删除。当更新广告的时候，把广告的原来的分类和新的分类的缓存删除。当删除广告，同样把广告所属的类的缓存删除。

## 购物车与订单模块

### 购物车实现

#### 购物车的实现

根据购物车的结构，创建一个Cart对象。

当用户未登录的时候，把商品的数据，添加到Cart对象，然后保存Cookie。需要展示购物车的时候，从cookie中取到数据。

当用户已经登录的时候，如果cookie中包含购物车信息，我们就把合并购物车（把cookie中的数据保存到redis中，并删除cookie中的数据）。再添加商品的数据，添加到Cart对象，然后保存到redis。需要展示购物车的时候，从Redis中取到数据。

### 订单实现

#### 商品详细页跨域请求

在商品详细页添加商品到购物车，这里需要实现跨域操作。所谓的跨域操作，就是使用js请求不同的域。不同的域是指协议，域名，端口，任一不同。

我们解决跨域使用CORS解决，也就是跨域资源共享。所谓实现跨域资源共享就是添加了预请求和预响应。在客户端angluars的请求中，设置withCredentials带有允许访问域，在服务端的response中设置Access-Control-Allow-Origin（允许访问的域）。

#### 收件人地址显示，默认，选择

使用angluarJS根据用户的user\_id,去查询tb\_address表。在页面中循环展示。当某条记录的is\_default属性是1的时候，显示为默认信息。当我们选择某一个地址的时候，这个地址为选中地址，显示选中样式。

#### 商品清单与金额显示

调用查询购物车，显示信息。

#### 提交订单

因为订单数据是分布式的数据库，所以需要使用分布式id生成器--雪花算法，实现全球唯一id。把订单信息保存到tb\_order表，把订单里面的订单项循环保存到tb\_order\_item表中。

如果订单是多个商家，我们保存到订单表也应该是多条记录。

当跳转订单以后，根据支付方式，如果支付是在线支付，就跳转到支付页面。否则，货到付款，就完成了整个页面的购物流程。

## Nginx

我们在做这个项目时，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.

(主动说)

当时我们使用nginx+3个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。

##### 怎么避免nginx产生单点故障

同时我们为了避免nginx的单点故障，达到高可用性，就在nginx的前面又加了一个F5，从而将请求分配给多个nginx，再通过nginx分配给多个不同的Tomcat。这样大大的提高了服务的有效性，并且进一步提高了性能。

nginx+tomcat负载均衡的配置

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并配置要转发到多台tomcat及其及其端口号，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）

3.在server标记下编写location规则信息，将css,js,html这些静态资源交给nginx处理，将动态的资源如以.do,.jsp结尾的通过proxy\_pass http://backend交给后端的tomcat集群处理，这样做的目的也是为了实现动静分离。

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的端口后一致即可。

##### 负载均衡中session共享具体是怎么做的

1.下载并安装redis服务

2.下载redis和tomcat融合时所需要的jar包,将其放到各个tomcat的lib文件夹下

3.配置各个tomcat中的context.xml文件和redis服务器进行通信

## 单点登录

#### 什么是单点登录（SSO）

在分布式系统中，在任何一个系统中登录，都可以访问其他的分布式系统，不需要重复登录。

CAS是解决单点登录的一种实现。

#### SSO单点登录访问流程主要步骤

首先SSO客户端把用户请求重定向到SSO服务端，SSO服务端实现用户身份认证，如果认证通过，SSO服务端会产生一个随机的Service Ticket,发送给用户。当用户访问其他系统的时候，去SSO客户端验证票据，如果验证通过，允许用户访问。

#### CAS的配置和测试

Cas的服务端就是一个war包。把服务端传到tomcat服务器上，访问http://IP:8080/cas/login，就是cas的登录。默认用户名和密码casuser /Mellon。

如果要修改cas的端口，首先需要修改tomcat的端口（conf/server.xml）,然后修改cas项目中的WEB-INF/cas.properties下的端口。

修改访问的协议，就是修改WEB-INF下的三个配置文件。

修改数据源web-inf下deployerConfigContext.xml，这里配置数据源，密码加密方式和数据库认证处理器。

测试：在web.xml中配置了登录过滤器，验证过滤器，登出过滤器和登出监听器。当我们访问登录页面，输入用户名和密码，cas实现认证。如果认证成功，默认跳转到index.jsp页面。

让我访问其他系统的时候，就不需要再次登录。登出就是访问[http://IP地址:端口/cas/logout](http://localhost:9000/cas/logout)

#### CAS客户端与SpringSecurity集成

首先需要cas依赖，springsecurity的依赖，还有cas和springsecurity的整合依赖。

然后在spring-security.xml文件中，配置入口点，指出cas实现用户的认证。指定登录的url，

认证过滤器，认证管理器，认证提供者，认证类，票据验证器，登出的监听器和地址。其中，

认证类实现了UserDetailsService接口，并重写了loadUserByUsername方法，这里我们只是实现用户的权限，而不没有实现用户的认证。

# 面试题的基本

### Java设计模式三种工厂模式

一．简单工厂模式

概述：简单工厂模式的创建意图就是，把对类的创建初始化全都交给一个工厂来执行，而用户不需要去关心创建的过程是什么样的，只用告诉工厂我想要什么就行了。而这种方法的缺点也很明显，违背了设计模式的开闭原则，因为如果你要增加工厂可以初始化的类的时候，你必须对工厂进行改建。

二. 工厂方法模式

为了解决简单工厂的问题，程序员们又想出来一个新的办法，就是设计一个工厂的接口，你想要什么东西，就写个类继承于这个工厂，这样就不用修改什么，直接添加就行了。

三. 抽象工厂模式

定义：为创建一组相关或相互依赖的对象提供一个接口，而且无需指定他们的具体类。

## SQL优化

外键必须加索引。

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。

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。

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。

用NOT EXISTS 替换 NOT IN 操作符，用 EXISTS 替换 IN

## java代码优化

解析大文件的xml数据使用sax替代dom4j,使用分段批量提交来完成大数据量的插入。

对于大批量字符串的拼接使用stringbuffer或者stringbuilder代替string进行+拼接。

根据业务情况使用缓存减少对数据库的访问。 单线程应尽量使用 HashMap, ArrayList，因为HashTable,Vector使用了同步机制，降低了性能。在finally块中关闭流,断开连接，释放资源。避免在循环条件中使用复杂表达式 。

### 方法重载和重写的区别

答：

1.重写必须继承，重载不用。  
2.重写的方法名，参数数目相同，参数类型兼容，重载的方法名相同，参数列表不同。  
3.重写的方法修饰符[大于等于](https://www.baidu.com/s?wd=%E5%A4%A7%E4%BA%8E%E7%AD%89%E4%BA%8E&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1dBryN9mHIBnyR1n19brAR0IAYqnWm3PW64rj0d0AP8IA3qPjfsn1bkrjKxmLKz0ZNzUjdCIZwsrBtEXh9GuA7EQhF9pywdQhPEUiqkIyN1IA-EUBtdnHc4nWR3rf" \t "_blank)父类的方法，重载和修饰符无关。  
4.重写不可以抛出父类没有抛出的一般异常，可以抛出运行时异常

### 单例手写懒汉式饿汉式

java单例模式是一种常见的设计模式，[单例模式](https://baike.baidu.com/item/%E5%8D%95%E4%BE%8B%E6%A8%A1%E5%BC%8F/5946627" \t "https://baike.baidu.com/item/java%E5%8D%95%E4%BE%8B%E6%A8%A1%E5%BC%8F/_blank)分三种：懒汉式单例、饿汉式单例、登记式单例三种。

单例模式有以下特点：  
　　1、单例类只能有一个实例。  
　　2、单例类必须自己创建自己的唯一实例。  
　　3、单例类必须给所有其他对象提供这一实例。

懒汉式

1. //懒汉式单例类.在第一次调用的时候实例化自己

2. public class Singleton {

3.     private Singleton() {}

4.     private static Singleton single=null;

5.     //静态工厂方法

6.     public static Singleton getInstance() {

7.          if (single == null) {

8.              single = new Singleton();

9.          }

10.         return single;

11.     }

12. }

饿汉式

1. //饿汉式单例类.在类初始化时，已经自行实例化

2. public class Singleton1 {

3.     private Singleton1() {}

4.     private static final Singleton1 single = new Singleton1();

5.     //静态工厂方法

6.     public static Singleton1 getInstance() {

7.         return single;

8.     }

9. }

### 泛型

**泛型，即“参数化类型”。一提到参数，最熟悉的就是定义方法时有形参，然后调用此方法时传递实参。那么参数化类型怎么理解呢？顾名思义，就是将类型由原来的具体的类型参数化，类似于方法中的变量参数，此时类型也定义成参数形式（可以称之为类型形参），然后在使用/调用时传入具体的类型（类型实参）。**

**泛型类型在逻辑上看以看成是多个不同的类型，实际上都是相同的基本类型。**

### 数据库分页

Mysql分页采用limt关键字 select \* from t\_order limit 5,10; #返回第6-15行数据 select \* from t\_order limit 5; #返回前5行 select \* from t\_order limit 0,5; #返回前5行 Mssql 2000分页采用top关键字(20005以上版本也支持关键字row[num](https://www.baidu.com/s?wd=num&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://zhidao.baidu.com/question/_blank)) Select top 10 \* from t\_order where id not in (select id from t\_order where id>5 ); //返回第6到15行数据 其中10表示取10记录 5表示从第5条记录开始取

Oracle分页①采用row[num](https://www.baidu.com/s?wd=num&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://zhidao.baidu.com/question/_blank)关键字(三层嵌套) SELECT \* FROM( SELECT A.\*,ROW[NUM](https://www.baidu.com/s?wd=NUM&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://zhidao.baidu.com/question/_blank) num FROM (SELECT \* FROM t\_order)A WHERE ROWNUM<=15) WHERE num>=5;--返回第5-15行数据 ②采用row\_number解析函数进行分页(效率更高) SELECT xx.\* FROM( SELECT t.\*,row\_number() over(ORDER BY o\_id)[AS](https://www.baidu.com/s?wd=AS&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://zhidao.baidu.com/question/_blank) num FROM t\_order t )xx WHERE num BETWEEN 5 AND 15; --返回第5-15行数据 解析函数能用格式 函数() over(pertion by 字段 order by 字段); Pertion 按照某个字段分区 Order 按照勒个字段排序 用分析函数 一万条后翻最后一页会很慢,使用ROWNUM 一万条之后每翻一页都会慢

### SSM和SSH框架优缺点

SpringMVC与Struts :

首先两者有个共同之处，那就是两者都数据javaweb层的开发框架，都是mvc模式的的经典产品，都实现了页面分离控制的功能，但是两者之间是有区别的。

有一种言论，或许已经成为了部分人眼中的事实，那就是SpringMVC已经超越了Struts，在开发中，人们更愿意使用SpringMVC而不是Struts。

因为Spring MVC属于SpringFrameWork的后续产品，已经融合在SpringWeb Flow里面。Spring MVC 分离了控制器、模型对象、分派器以及处理程序对象的角色，这种分离让它们更容易进行定制。

而且由于同是属于一个开发公司，所以SpringMVC比Struts更容易契合Spring技术，在扩展和灵活性上更胜一筹。

Struts的优势在于静态注入，插件机制和拦截器链，但是struts存在漏洞，经常会被作为攻击点进行冲击。相比更加安全简单的SpringMVC，开发者渐渐开发放弃了它。

Hibernate 与MyBatis:

Hibernate与Mybatis都是流行的持久层开发框架，一句话概括：MyBatis 简单易上手；hibernate成熟，市场推广率高。

MyBatis可以进行更为细致的SQL优化，可以减少查询字段。

MyBatis容易掌握，而Hibernate门槛较高。

更重要的是，mybatis提供了对应各种用途、功能的插件，有需求？好，来个插件就搞定。而hibernate在这一方面是远远比不上mybatis的。

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

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。

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。

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。

但是hibernat缺点很明确，如果涉及到多张关联表的调用时：

1. 多表关联等比较复杂，使用的成本并不低；

2. 效率比较低，在大型项目中很少会使用到它，因为sql都是自动生成的，不太好进行人工的优化。

### Jdk1.7的特性

switch语句中支持使用字符串了

支持 List tempList = new ArrayList<>()的声明方式，其实是泛型实例化类型自动推断。

提供自定义关闭类的接口，实现AutoCloseable ，就可以在类销毁的时候自动关闭一些资源。

char值可以进行equals比较

boolean值支持反转和位运算

File System.\* 增加了很多get环境信息的工具方法

try catch支持捕获多个异常，竖线分割异常即可

try块中使用的资源可以不用手动在finally中关闭

安全的加减乘除操作

### StringBoot

#### SpringBoot特性

##### 二、springboot主要特性

1、遵循习惯优于配置的原则。使用springboot我们只需要很少的配置，大多数使用默认配置即可

2、项目快速搭建。springboot帮助开发者快速搭建spring框架，可无需配置的自动整合第三方框架

3、可以完全不使用xml配置，只需要自动配置和Java config

4、内嵌servlet容器，降低了对环境的要求，可用命令直接执行项目

5、提供了starter POM，能够非常方便的进行包管理

6、对主流框架无配置集成

7、与云计算天然集成

##### 三、springboot的核心功能

1、独立运行spring项目。springboot可以以jar包的形式进行独立运行，使用java -jar xx.jar 就可以成功运行项目。

2、内嵌servlet容器。内嵌容器，使得我们可以执行运行项目的主程序main函数，使得项目可以快速运行。

3、提供starter简化Maven配置。springboot提供了一系列的start pol用来简化maven依赖。如：常用的spring-boot-starter-web、spring-boot-starter-tomcat、spring-boot-starter-actuator等

4、自动配置spring。springboot会根据我们项目中类路径的jar包，为jar包的类进行自动装配bean。

5、应用监控。springboot提供了基于HTTP、ssh、telnet对运行时的项目进行监控。

springboot提供了actuator组件，只需要在配置中加入spring-boot-starter-actuator依赖，通过继承AbstractHealthIndicator这个抽象类，然后在doHealthCheck()方法中检测服务健康的方法，就可以实现一个简单的监控。

##### 四、springboot的缺点

1、高度集成，使用的时候不知道底层实现

2、由于不了解底层，导致项目出了问题会很难排查

##### SpringBoot几个常用的注解

（1）@RestController和@Controller指定一个类，作为控制器的注解 ，并说明其区别

（2）@RequestMapping方法级别的映射注解，这一个用过Spring MVC的小伙伴相信都很熟悉

（3）@EnableAutoConfiguration和@SpringBootApplication是类级别的注解，根据maven依赖的jar来自动猜测完成正确的spring的对应配置，只要引入了spring-boot-starter-web的依赖，默认会自动配置Spring MVC和tomcat容器

（4）@Configuration类级别的注解，一般这个注解，我们用来标识main方法所在的类,完成元数据bean的初始化。

（5）@ComponentScan类级别的注解，自动扫描加载所有的Spring组件包括Bean注入，一般用在main方法所在的类上

（6）@ImportResource类级别注解，当我们必须使用一个xml的配置时，使用@ImportResource和@Configuration来标识这个文件资源的类。

（7）@Autowired注解，一般结合@ComponentScan注解，来自动注入一个Service或Dao级别的Bean

（8）@Component类级别注解，用来标识一个组件，比如我自定了一个filter，则需要此注解标识之后，Spring Boot才会正确识别。

springboot是如何管理事务的？直接写@transactional注解就行，在方法就是方法事务，类上就是类事务

### FastDFS

#### FastDFS是什么？

FastDFS 是一个开源的高性能分布式文件系统（DFS）。 它的主要功能包括：文件存储，文件同步和文件访问，以及高容量和负载平衡。

#### FastDFS角色

FastDFS 系统有三个角色：**跟踪服务器(Tracker Server)、存储服务器(Storage Server)和  
客户端(Client)**。

**Tracker Server**: 跟踪服务器，主要做调度工作，起到均衡的作用；负责管理所有的 storage server  
和 group，每个 storage 在启动后会连接 Tracker，告知自己所属 group 等信息，并保持周期性心跳。

**Storage Server**：存储服务器，主要提供容量和备份服务；以 group 为单位，每个 group 内可以有多台 storage server，数据互为备份。

**Client**:客户端，上传下载数据的服务器，也就是我们自己的项目所部署在的服务器。

### Redis

#### Redis的特性

1、速度快

Redis是用C语言实现的；

Redis的所有数据存储在内存中。

2、持久化

Redis的所有数据存储在内存中，对数据的更新将异步地保存到磁盘上。

3、支持多种数据结构

Redis支持五种数据结构：String、List、Set、Hash、Zset

4、支持多种编程语言

Java、php、Python、Ruby、Lua、Node.js

5、功能丰富

除了支持五种数据结构之外，还支持事务、流水线、发布/订阅、消息队列等功能。

6、源码简单

约23000行C语言源代码。

7、主从复制

主服务器（master）执行添加、修改、删除，从服务器执行查询。

1. 高可用及分布式 。

#### Redis穿透和雪崩

redis穿透：正常的执行路径是这样的，请求数据，首先会从redis缓存中拿数据，如果缓存没有的话才去查数据库，再写到redis缓存中。那么如果有人请求一条根本不存在的数据时，redis里面肯定没有嘛，它就会去访问数据库，但是数据库没有，所以它也没把数据写回redis缓存。所以它每次请求这个数据的时候它就会直接去访问数据库。如果请求的数量太大的话，都直接穿过redis直接去访问数据库，数据库承受不了这个访问数量。

解决办法：在redis里面用一个set集来把数据库查询的那个查询主键都读出来存到这个set集里面。如果有请求时，先查redis里面的set有没有这个主键，如果没有就直接返回，不查数据库。如果有的话查redis，如果没有的话才去查数据库并把数据库里的数据写到缓存中。

redis雪崩：每个key（即数据）如果设置了失效时间的话，如果大量key同时过期的时候，这些key又大量地去请求这些key时，因为redis里面没有这些数据，就会大量的请求就会大量涌向数据库，就会导致数据库处理不过来，导致“雪崩”。

解决办法：在设置失效时间的时候，给它加一个随机的秒数（0~60），来让这些大量的数据进行错开对数据库的访问。这样数据库就能应付过来了。如果这个key的访问频率频繁的时候，我们可以让它每查一次就给它加点有效时间。这样就能解决雪崩问题了

### 谈谈你对Spring 的理解：

#### IOC

.控制反转(IOC)：以前传统的java开发模式中，当需要一个对象时我们，我们会自己使用new或者getInstance等直接或者间接调用构造方法创建一个对象，而在Spring开发模式中，Spring容器使用了工厂模式为我们创建了所需要的对象，我们使用时不需要自己去创建，直接调用Spring为我们提供的对象即可，这就是控制反转的思想。实例化一个java对象有三种方式：使用类构造器，使用静态工厂方法，使用实例工厂方法，当使用spring时我们就不需要关心通过何种方式实例化一个对象，spring通过控制反转机制自动为我们实例化一个对象。IOC把对象的创建、初始化、销毁交给spring来管理，而不是由开发者控制，实现控制反转。

#### .依赖注入(DI)

Spring使用java Bean对象的Set方法或者带参数的构造方法为我们在创建所需对象时将其属性自动设置所需要的值的过程就是依赖注入的基本思想。

.**面向切面编程(AOP)**：

在面向对象编程(OOP)思想中，我们将事物纵向抽象成一个个的对象。而在面向切面编程中，我们将一个个对象某些类似的方面横向抽象成一个切面，对这个切面进行一些如权限验证，事物管理，记录日志等公用操作处理的过程就是面向切面编程的思想。（一句话概括：把切面中的通知织入到目标对象的连接点的相应位置，通过aop代理实现）

AOP即面向切面编程（对象是接口），实现AOP功能采用的是代理技术（jdk针对接口代理；cglib针对类实现代理），客户端程序不再调用目标，而调用代理类，代理类与目标类对外具有相同的方法声明；AOP适合于那些具有横切逻辑的应用：如性能监测，访问控制，事务管理以及日志记录。 从Spring的角度看，AOP最大的用途就在于提供了事务管理的能力。Spring在你访问数据库之前，自动帮你开启事务，当你访问数据库结束之后，自动帮你提交/回滚事务！

#### 事务的七种传播特性

（1）propagation\_requierd：如果当前没有事务，就新建一个事务，如果已存在一个事务中，加入到这个事务中，这是最常见的选择。

（2）propagation\_supports：支持当前事务，如果没有当前事务，就以非事务方法执行。

（3）propagation\_mandatory：使用当前事务，如果没有当前事务，就抛出异常。

（4）propagation\_required\_new：新建事务，如果当前存在事务，把当前事务挂起。（即使存在合适的事务 也要[新建](https://www.baidu.com/s?wd=%E6%96%B0%E5%BB%BA&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1dBuHb3P1KWPvFhrA7WPWF-0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPHRznWR3PHcsnj6dPHbYrjDz)一个来使用）

（5）propagation\_not\_supported：以非事务方式执行操作，如果当前存在事务，就把当前事务挂起。

（6）propagation\_never：以非事务方式执行操作，如果当前事务存在则抛出异常。

（7）propagation\_nested：如果当前存在事务，则在嵌套事务内执行。如果当前没有事务，则执行与propagation\_required类似的操作

#### Spring事务的四种隔离级别(未提交读，提交读，可重复读，串行读)

#### **未提交读(Read Uncommitted)： 3不防**

允许脏读，也就是可能读取到其他会话中未提交事务修改的数据。

#### **提交读(Read Committed)： 防脏**

只能读取到已经提交的数据。Oracle等多数数据库默认都是该级别（不重复读）

#### **可重复读(**Repeatable **Read)：防脏 防不可重复读**

可重复读。在同一个事务内的查询都是师傅开始时刻一致的，InnoDB默认级别。在SQL标准中，该隔离级别消除了不可重复读，但是还存在幻读。

#### 串行读(Serializable)：3防

完全串行化的读，每次读都需要获得表级共享锁，读写相互都会阻塞

#### **其中的一些概念的说明：**

脏读: 指当一个事务正在访问数据，对数据进行了修改，而修改还没有提交到数据库中，这时，另外一个事务也访问这个数据，然后使用了这个数据。因为这个数据是还没有提交的数据， 那么另外一个事务读到的这个数据是脏数据，依据脏数据所做的操作可能是不正确的。

不可重复读: 指在一个事务内，多次读同一数据。在这个事务还没有结束时，另外一个事务也访问该同一数据。 那么，在第一个事务中的两次读数据之间，由于第二个事务的修改，那么第一个事务两次读到的数据可能是不一样的。这样就发生了在一个事务内两次读到的数据是不一样的，因此称为是不可重复读。

幻读: ，例如第一个事务对一个表中的数据进行了修改，这种修改涉及到表中的全部数据行。同时，第二个事务也修改这个表中的数据，这种修改是向表中插入一行新数据。那么，以后就会发生操作第一个事务的用户发现表中还有没有修改的数据行，就好象发生了幻觉一样。

### 谈谈你对SpringMVC的理解

原理：SpringMvc是基于过滤器对servlet进行了封装的一个框架，我们使用的时候就是在web.xml文件中配置DispatcherServlet类；SpringMvc工作时主要是通过DispatcherServlet管理接收到的请求并进行处理。

**SpringMVC话术:**

当用户发送一个请求到了前端控制器DispatcherServlet之后，前端控制器DispatcherServlet会去调用HandlerMapping去查找对应的处理器，然后处理器在处理请求时会调用相应的javabean进行业务数据的封装和业务逻辑处理，执行完成之后会把ModelAndView返回给DispatcherServlet，然后前端控制器DispatcherServlet通过视图解析器ViewReslover进行解析，解析之后返回具体的View，然后前端控制器DispatcherServlet根据View进行渲染视图，最后响应给用户！

### 谈谈你对MVC的理解

MVC是Model—View—Controler的简称。即模型—视图—控制器。MVC是一种设计模式，它强制性的把应用程序的输入、处理和输出分开。

MVC中的模型、视图、控制器它们分别担负着不同的任务。

视图: 视图是用户看到并与之交互的界面。视图向用户显示相关的数据，并接受用户的输入。视图不进行任何业务逻辑处理。

模型: 模型表示业务数据和业务处理。相当于JavaBean。一个模型能为多个视图提供数据。这提高了应用程序的重用性

控制器: 当用户单击Web页面中的提交按钮时,控制器接受请求并调用相应的模型去处理请求。然后根据处理的结果调用相应的视图来显示处理的结果。

MVC的处理过程：首先控制器接受用户的请求，调用相应的模型来进行业务处理，并返回数据给控制器。控制器调用相应的视图来显示处理的结果。并通过视图呈现给用户。

### SpringMVC有哪些注解

@Controller声明此类是一个控制层类，后续dispatchServlet在分发请求时就可以请求到该类，@Controller的注解需要与 @RequestMapping结合使用才能访问到控制层的具体方法。

@Service声明此类是一个业务逻辑层类

@RequestMapping结合使用，整体使用感觉和@requestParam差别不大。

@ResponseBody 把返回的参数转换成xml或者json格式，主要与ajax请求结合使用。

@RequestBody 把request请求中的json参数转为对应的实体对象，要求必须在请求头信息中声明参数类型contentType:"application/json。

@Resource默认按照ByName（根据名字）自动注入，由J2EE提供，需要导入包javax.annotation.Resource。@Resource有两个重要的属性：name和type，而Spring将@Resource注解的name属性解析为bean的名字，而type属性则解析为bean的类型。所以，如果使用name属性，则使用byName的自动注入策略，而使用type属性时则使用byType自动注入策略。如果既不制定name也不制定type属性，这时将通过反射机制使用byName自动注入策略。

@Autowired为Spring提供的注解，需要导入包org.springframework.beans.factory.annotation.Autowired;只按照byType注入（根据类型注入）。@Autowired注解是按照类型（byType）装配依赖对象，默认情况下它要求依赖对象必须存在，如果允许null值，可以设置它的required属性为false。如果我们想使用按照名称（byName）来装配，可以结合@Qualifier注解一起使用。

### Mybatis理解

#### 1.mybatis是什么

mybatis是一个优秀的持久层框架，他对jdbc操作数据库的过程进行了封装，使开发着只用关注sql本身，不用去关注例如注册驱动，加载链接，得到statement，处理结果集等复杂的过程。

mybatis通过xml或者注解的方式，将要执行的各种sql语句配置起来，并通过Java对象和statement中的sql语句映射生成最终的sql语句，最后由mybatis框架执行sql语句，并将结果映射成Java对象返回。

#### 2.工作原理

mybatis通过配置文件创建sqlsessionFactory，通过sqlsessionFactory，获取sqlSession。SQLSession包含了执行sql语句的所有方法，可以通过SQLSession直接运行映射的sql语句，完成对数据的增删改查和事物的提交工作，用完之后关闭SQLSession。

#### 3.mybatis解决的问题

1）.使用数据库连接池管理链接，避免了频繁创建了、关闭链接，浪费资源，影响性能的问题。

2）.用xml管理sql语句，让Java代码和sql语句分离，使得代码更易维护。

3）.解决了sql语句参数不定的问题。xml中可以通过where条件决定sql语句的条件参数。mybatis将Java对象映射到sql语句，通过statement的parameterType定义输入参数的类型。

4）.mybatis自动将结果集封装成Java对象， 通过statement的resultType定义输出的类型。避免了因sql变化，对结果集处理麻烦的问题。

#### 4.mybatis配置延迟加载（懒加载）

概念:延迟加载即先从单表查询、需要时再从关联表去关联查询，大大提高数据库性能，因为查询单表要比关联查询多张表速度要快。

开发流程:

在mybatis的核心配置文件当中配置settings 设置懒加载属性,注意:在properties标签后面加入下面标签

<settings>

//打开延迟加载的开关

<setting name="lazyLoadingEnabled" value="true"/>

//将积极加载改为消极加载，即延迟加载

<setting name="aggressiveLazyLoading" value="false"/>

</settings>

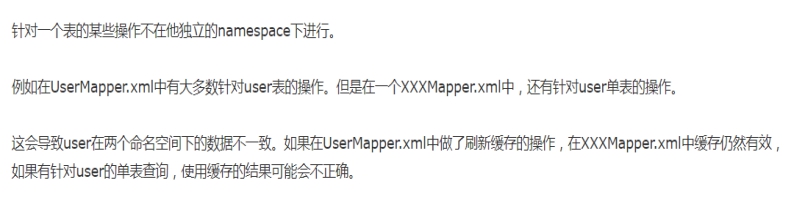
使用association标签完成一对一映射时，select属性指定延迟加载需要执行的statement(必须指定该statement的唯一标识),这样即完成了延迟加载.

#### 5.Mabatis的缓存

**MyBatis有两级缓存：**

一级缓存：是SqlSession级别的缓存，是基于 HashMap 的本地缓存。不同的 SqlSession 之间的缓存数据区域互不影响。他的作用域是同一个SqlSession，当同一个 SqlSession 执行两次相同的 sql 语句时，第一次执行完后会将数据库中查询的数据写到缓存，第二次查询时直接从缓存获取不用去数据库查询。当 SqlSession 执行 insert、update、delete 操做并提交到数据库时，会清空缓存，保证缓存中的信息是最新的。

二级缓存：是mapper级别的缓存，作用域是mapper的同一个namespace， 不同的sqlsession在同一个namespace下执行相同的sql语句，第一次查询的结果缓存到缓存中，第二次则可从缓存中获取数据。

**二级缓存危害：**

Mybatis默认开启一级缓存，没有开启二级缓存，二级缓存需要在xml配置中手动配置开启。

### Hibernate5个核心接口

1，Configuration接口：配置Hibernate，根据其启动Hibernate，创建SessionFactory对象；

2，SessionFactory接口：初始化Hibernate，充当数据存储源的代理，创建session对象，SessionFactory是

线程安全的，意味着它的同一个实例可以被应用的多个线程共享，是重量级二级缓存；

3，session接口：负责保存、更新、删除、加载和查询对象，是一个非线程安全的，避免多个线程共享一个session，是轻量级，一级缓存。

4，Transaction接口：管理事务。可以对事务进行提交和回滚；

5，Query和Criteria接口：执行数据库的查询。

### Hibernate中一级缓存，二级缓存，查询缓存表示什么？

1. Session缓存（又称作事务缓存）：Hibernate内置的，不能卸除。  
     
     
     
     
   缓存范围：缓存只能被当前Session对象访问。缓存的生命周期依赖于Session的生命周期，当Session被关闭后，缓存也就结束生命周期。  
     
     
   Hibernate一些与一级缓存相关的操作（时间点）：  
   数据放入缓存：  
   1. save()。当session对象调用save()方法保存一个对象后，该对象会被放入到session的缓存中。  
   2. get()和load()。当session对象调用get()或load()方法从数据库取出一个对象后，该对象也会被放入到session的缓存中。  
   3. 使用HQL和QBC等从数据库中查询数据。  
   二、SessionFactory缓存（又称作应用缓存）：使用第三方插件，可插拔。  
   缓存范围：缓存被应用范围内的所有session共享,不同的Session可以共享。这些session有可能是并发访问缓存，因此必须对缓存进行更新。缓存的生命周期依赖于应用的生命周期，应用结束时，缓存也就结束了生命周期，二级缓存存在于应用程序范围。  
   适合放到二级缓存中：  
   （1）经常被访问  
   （2）改动不大  
   （3）数量有限  
   （4）不是很重要的数据，允许出现偶尔并发的数据。   
   这样的数据非常适合放到二级缓存中的。  
   用户的权限：用户的数量不大，权限不多，不会经常被改动，经常被访问。

### Map整理

###### 常见问题：

1、“你知道HashMap的工作原理吗？” “你知道HashMap的get()方法的工作原理吗？”

　　HashMap是基于hashing的原理，我们使用put(key, value)存储对象到HashMap中，使用get(key)从HashMap中获取对象。当我们给put()方法传递键和值时，我们先对键调用hashCode()方法，返回的hashCode用于找到bucket位置来储存Entry对象。”这里关键点在于指出，HashMap是在bucket中储存键对象和值对象，作为Map.Entry。

###### 2、“当两个对象的hashcode相同会发生什么？”

　　因为hashcode相同，所以它们的bucket位置相同，‘碰撞’会发生。因为HashMap使用LinkedList存储对象，这个Entry(包含有键值对的Map.Entry对象)会存储在LinkedList中。(当向 HashMap 中添加 key-value 对，由其 key 的 hashCode() 返回值决定该 key-value 对（就是 Entry 对象）的存储位置。当两个 Entry 对象的 key 的 hashCode() 返回值相同时，将由 key 通过 eqauls() 比较值决定是采用覆盖行为（返回 true），还是产生 Entry 链（返回 false）。)

###### 3、“如果两个键的hashcode相同，你如何获取值对象？”

　　当我们调用get()方法，HashMap会使用键对象的hashcode找到bucket位置，然后获取值对象。如果有两个值对象储存在同一个bucket，将会遍历LinkedList直到找到值对象。找到bucket位置之后，会调用keys.equals()方法去找到LinkedList中正确的节点，最终找到要找的值对象。(当程序通过 key 取出对应 value 时，系统只要先计算出该 key 的 hashCode() 返回值，在根据该 hashCode 返回值找出该 key 在 table 数组中的索引，然后取出该索引处的 Entry，最后返回该 key 对应的 value 即可。)

###### 4、“如果HashMap的大小超过了负载因子(load factor)定义的容量，怎么办？”

　　当一个map填满了75%的bucket时候，和其它集合类(如ArrayList等)一样，将会创建原来HashMap大小的两倍的bucket数组，来重新调整map的大小，并将原来的对象放入新的bucket数组中。这个过程叫作rehashing，因为它调用hash方法找到新的bucket位置。

###### 5、“你了解重新调整HashMap大小存在什么问题吗？”

当重新调整HashMap大小的时候，确实存在条件竞争，因为如果两个线程都发现HashMap需要重新调整大小了，它们会同时试着调整大小。在调整大小的过程中，存储在LinkedList中的元素的次序会反过来，因为移动到新的bucket位置的时候，HashMap并不会将元素放在LinkedList的尾部，而是放在头部，这是为了避免尾部遍历(tail traversing)。如果条件竞争发生了，那么就死循环了。这个时候，你可以质问面试官，为什么这么奇怪，要在多线程的环境下使用HashMap呢？

##### Map的特征

（1）Map不是collection的子接口或者实现类。Map是一个接口。

Map的每个Entry都持有两个对象，也就是一个   
 键一个值，Map可能会持有相同的值对象但键对   
 象必须是唯一的。

TreeMap也通过Comparator或者Comparable  
 维护一个排序顺序。

Map里你可以拥有随意个null值但最多只能有  
 一个null值。

Map接口最流行的几个实现类是HashMap、  
 LinkedHashMap、Hashtable和TreeMap。   
 （HashMap、TreeMap最常用）

###### HashMap和Hashtable的区别

#### （1）两者最主要的区别在于Hashtable是线程安全，而HashMap则非线程安全

**Hashtable的实现方法里面都添加了synchronized关键字来确保线程同步，因此相对而言HashMap性能会高一些，我们平时使用时若无特殊需求建议使用HashMap，在多线程环境下若使用HashMap需要使用Collections.synchronizedMap()方法来获取一个线程安全的集合**（Collections.synchronizedMap()实现原理是Collections定义了一个SynchronizedMap的内部类，这个类实现了Map接口，在调用方法时使用synchronized来保证线程同步,当然了实际上操作的还是我们传入的HashMap实例，简单的说就是Collections.synchronizedMap()方法帮我们在操作HashMap时自动添加了synchronized来实现线程同步，类似的其它Collections.synchronizedXX方法也是类似原理）

#### （2）HashMap可以使用null作为key，而Hashtable则不允许null作为key

虽说HashMap支持null值作为key，不过建议还是尽量避免这样使用，因为一旦不小心使用了，若因此引发一些问题，排查起来很是费事   
HashMap以null作为key时，总是存储在table数组的第一个节点上

#### （3）HashMap是对Map接口的实现，HashTable实现了Map接口和Dictionary抽象类

#### （4）HashMap的初始容量为16，Hashtable初始容量为11，两者的填充因子默认都是0.75

###### （5）两者计算hash的方法不同

Hashtable计算hash是直接使用key的hashcode对table数组的长度直接进行取模

#### （7）HashSet和HashMap、Hashtable的区别

除开HashMap和Hashtable外，还有一个hash集合HashSet，有所区别的是HashSet不是key value结构，仅仅是存储不重复的元素，相当于简化版的HashMap，只是包含HashMap中的key而已

通过查看源码也证实了这一点，HashSet内部就是使用HashMap实现，只不过HashSet里面的HashMap所有的value都是同一个Object而已，因此HashSet也是非线程安全的，至于HashSet和Hashtable的区别，HashSet就是个简化的HashMap的

##### List、Map、Set三个接口，存取元素时，各有什么特点？

List与Set都是单列元素的集合，它们有一个功共同的父接口Collection。

###### Set里面不允许有重复的元素

存元素：add方法有一个boolean的返回值，当集合中没有某个元素，此时add方法可成功加入该元素时，则返回true；当集合含有与某个元素equals相等的元素时，此时add方法无法加入该元素，返回结果为false。

取元素：没法说取第几个，只能以Iterator接口取得所有的元素，再逐一遍历各个元素。

###### List表示有先后顺序的集合

存元素：多次调用add(Object)方法时，每次加入的对象按先来后到的顺序排序，也可以插队，即调用add(int index,Object)方法，就可以指定当前对象在集合中的存放位置。

取元素：方法1：Iterator接口取得所有，逐一遍历各个元素

方法2：调用get(index i)来明确说明取第几个。

Map是双列的集合，存放用put方法:put(obj key,obj value)，每次存储时，要存储一对key/value，不能存储重复的key，这个重复的规则也是按equals比较相等。

取元素：用get(Object key)方法根据key获得相应的value。

也可以获得所有的key的集合，还可以获得所有的value的集合，

还可以获得key和value组合成的Map.Entry对象的集合。

List以特定次序来持有元素，可有重复元素。Set 无法拥有重复元素,内部排序。Map 保存key-value值，value可多值。

# 如何看待你上家公司

每个公司都有自己的一个企业文化，从上一家公司我学到了很多东西，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。

# 离职原因

因为在上一家公司时间较长，想换个环境，告别先前的自己。