校园商城

一．技术栈

前端：html,js,css

后端：SSM redis mysql 图片处理

部署运营：阿里云

技术储备

JavaWeb Maven SSM

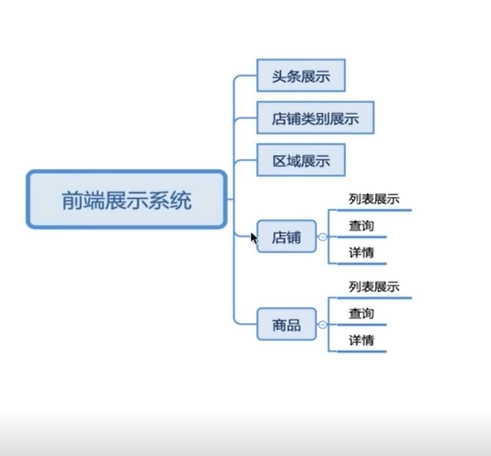
开发工具

JDK8 Maven3.3.9 Mysql5.5+ 浏览器 Tomcat8 IDEA

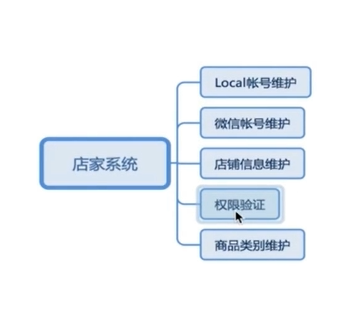
1. 创建Maven,web-app工程

二．系统划分

2.1前端展示系统



2.2店家系统



本地账号：用户自己注册的账号密码

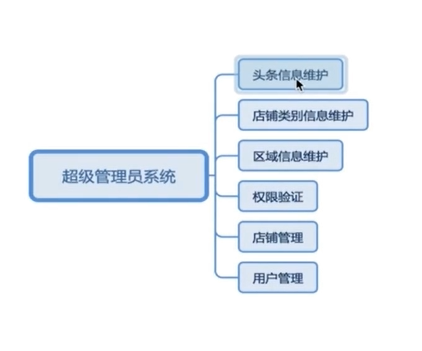
微信账号维护：支持微信登录

权限验证：验证是否有权限进行修改商品信息

店铺信息：管理店铺

商品类别维护：店家管理商品数目和上下架

2.3超级管理员系统



头条信息：o2o系统的首页

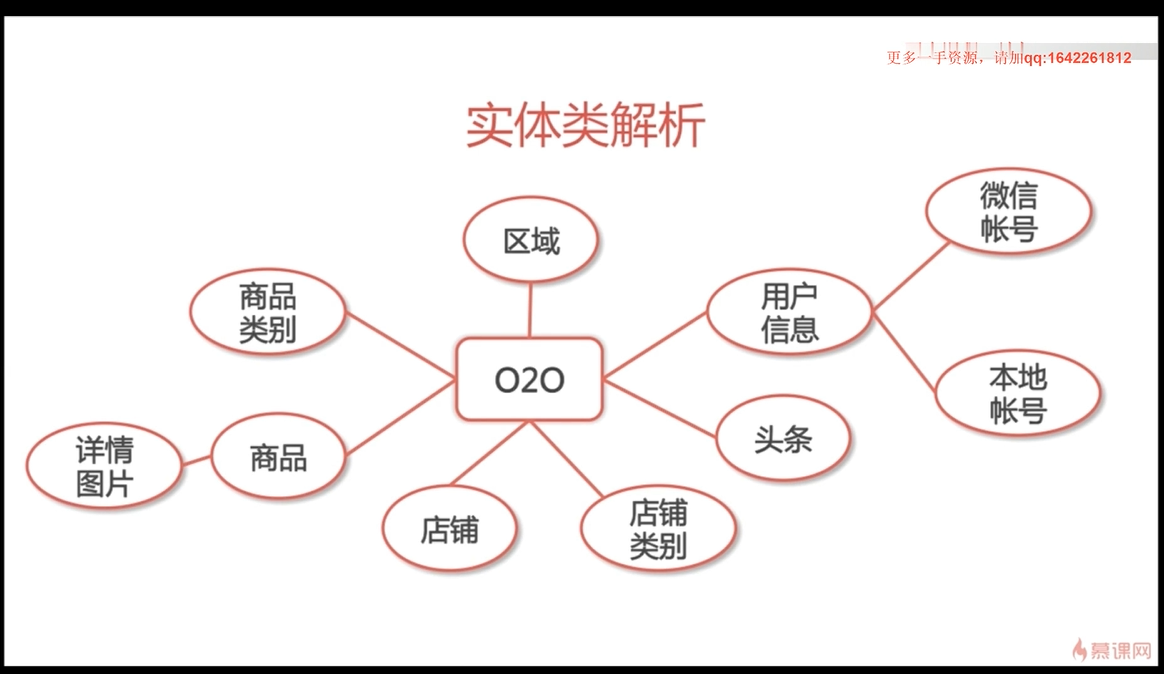
店铺类别：店铺分为哪几类

区域信息：店铺所在区域

权限：店家和客户都不能登录

店铺，用户管理：管理店铺（只有审核通过的才行，和店铺排序）和用户，处理禁用违规用户和店铺

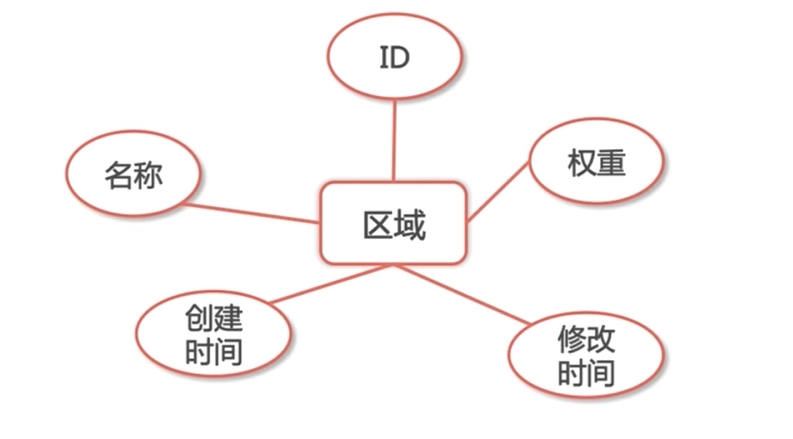
三：实体类(cn.xww. o2o.domain)



用户信息--微信账号—本地账号，通过实体类的用户id关联

3.1区域（店铺所在位置）

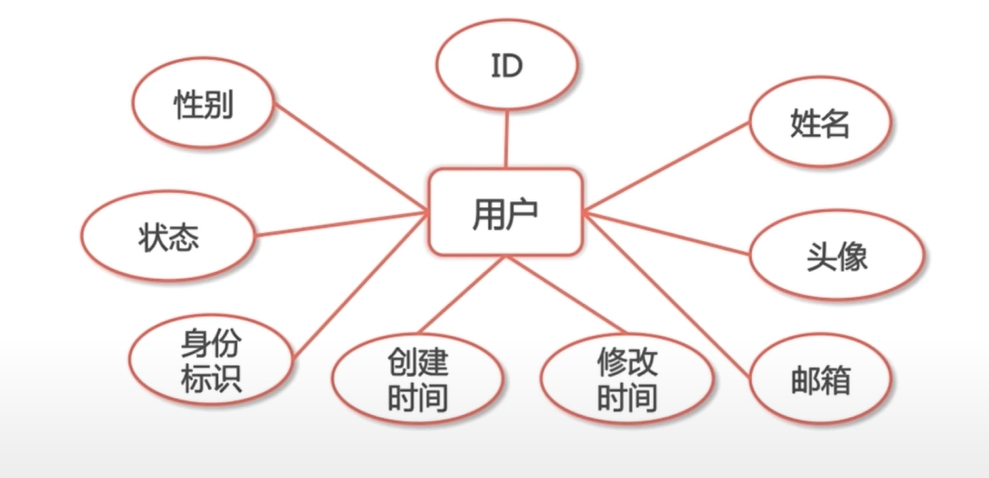
权重：区域展示的优先级



3.2用户类

状态：是否可登陆

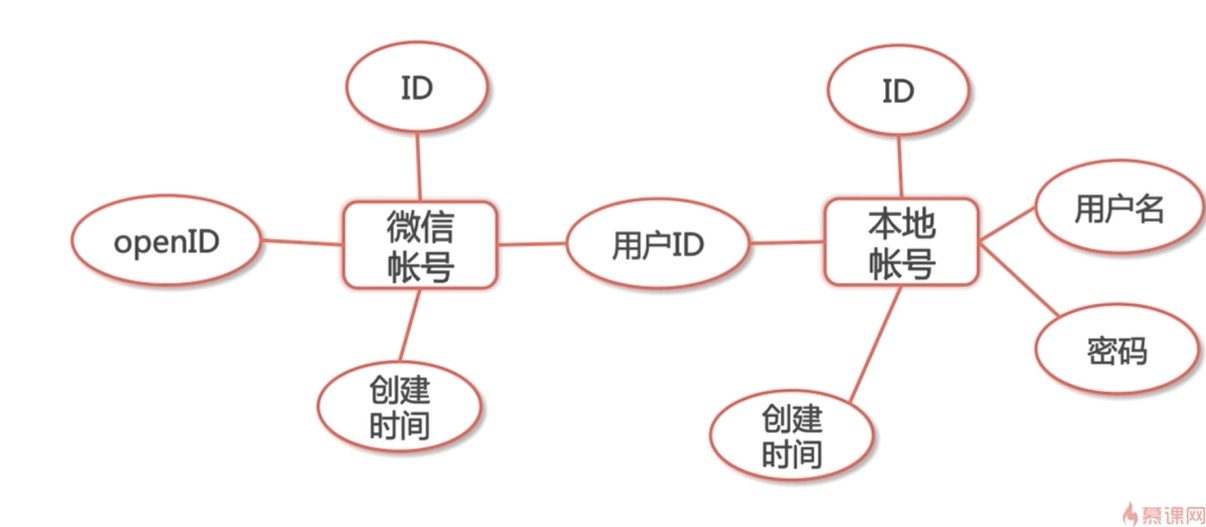
身份表示：用户，商家，超级管理员



3.3微信账号和本地账号

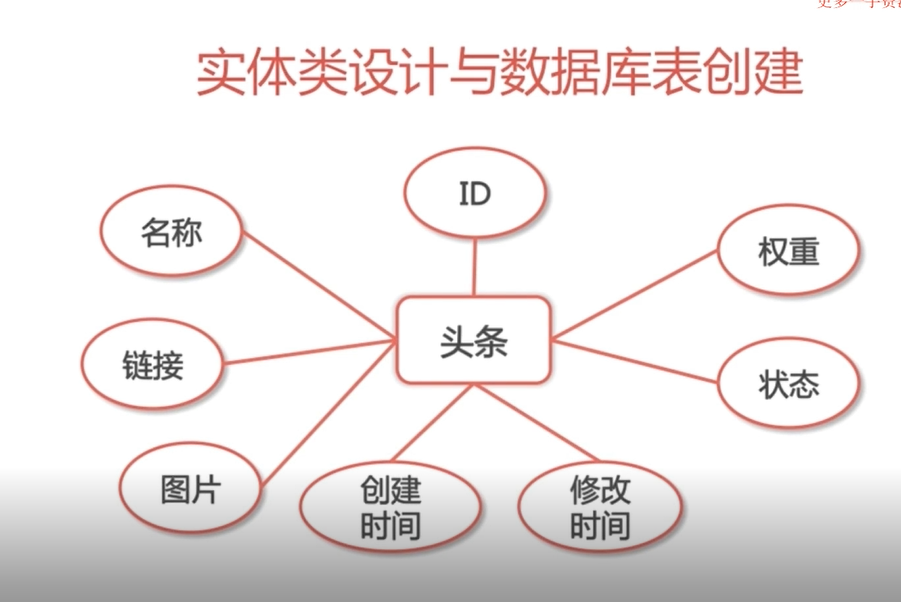
微信账号：绑定微信登录

本地账号：注册的账号



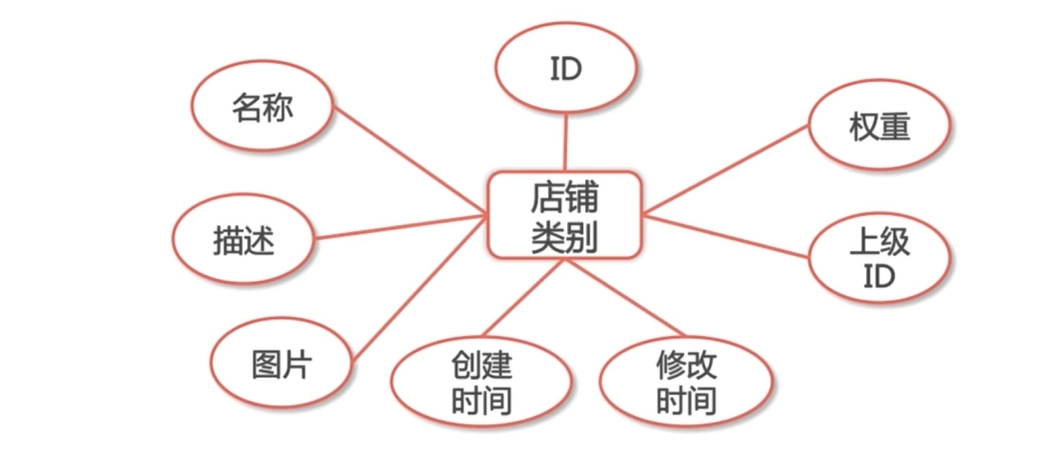
3.4头条

状态：是否可用



3.5店铺类别（店铺属于哪一类）

上级id：例如饮品—香飘飘；香飘飘的上级id就是饮品



3.6店铺

权重：越大展示的越靠前

区域ID：店铺属于哪一区域

用户ID：店铺拥有者（不是客户）

状态：是否可用（超级管理员是否审核通过）

建议：超级管理员的建议



3.7商品类别

Id

店铺id:是哪个店铺下的类别

商品类别名称

权重

创建时间

3.8商品图片详情

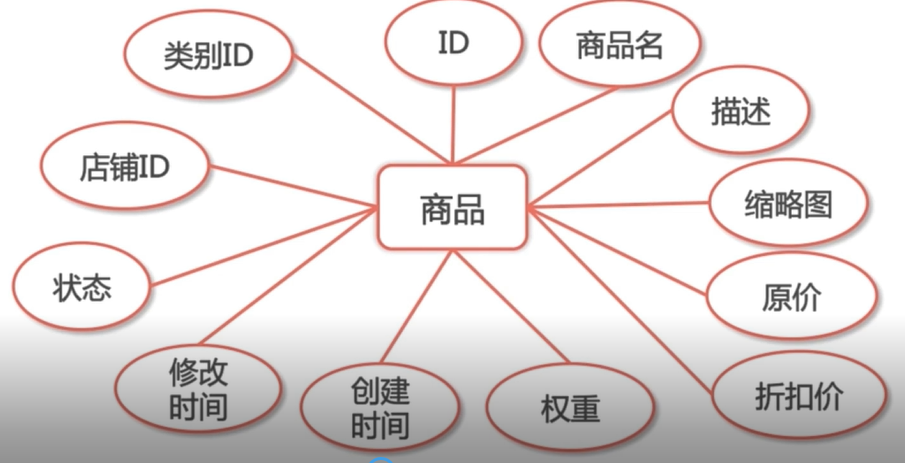
图片id;图片地址；图片描述；权重；创建时间；图片所属商品id

3.9商品

缩略图:商品图片地址

店铺id:该商品属于哪一个店铺

类别id:商品类别



四.步骤

4.1搭建环境（Maven下的web-app骨架）

4.2完善相应的包

4.3创建实体类

区域

用户

微信账号与本地账号（通过用户id关联，绑定在一起）

头条

店铺类别

店铺

商品类别

商品

商品图片详情

4.4数据库中构建实体类对应的表

区域

CREATE TABLE `tb\_area`(

`area\_id` INT(2) NOT NULL PRIMARY KEY AUTO\_INCREMENT,

`area\_name` VARCHAR(200) NOT NULL UNIQUE,

`priority` INT(2) NOT NULL DEFAULT '0',

`create\_time` DATETIME DEFAULT NULL,

`last\_edit\_time` DATETIME DEFAULT NULL

)ENGINE=INNODB AUTO\_INCREMENT=1 DEFAULT CHARSET=utf8;

注:1.mysql建立表格引用表键用`，插入数据用‘’

2. INT(2)表示显示2宽度，也就是0-99

3. ENGINE=INNODB mysql引擎，此为行集锁，也就是一行一个锁，支持多并发修改；MYISAM为表集锁，一张表一个锁，修改另外行必须等待当前行修改线程结束。

INNODB 支持事务，写的快

MYISAM 读的快

4. AUTO\_INCREMENT=1 按1自增长

用户

CREATE DATABASE o2o;

USE o2o;

CREATE TABLE `tb\_person\_info`(

`user\_id` INT(10) NOT NULL PRIMARY KEY AUTO\_INCREMENT,

`name` VARCHAR(32) DEFAULT NULL,

`profile\_img` VARCHAR(1024) DEFAULT NULL,

`email` VARCHAR(1024) DEFAULT NULL,

`gener` VARCHAR(2) DEFAULT NULL,

`enable\_status` INT(2) NOT NULL DEFAULT '0',

`user\_type` INT(2) NOT NULL DEFAULT '1',

`create\_time` DATETIME DEFAULT NULL,

`last\_edit\_time` DATETIME DEFAULT NULL

)ENGINE=INNODB AUTO\_INCREMENT=1 DEFAULT CHARSET=utf8;

微信账号

CREATE TABLE `tb\_wechat\_auth`(

`wechat\_auth\_id` INT(10) NOT NULL PRIMARY KEY AUTO\_INCREMENT,

`user\_id` INT(10) NOT NULL,

`create\_time` DATETIME DEFAULT NULL,

`open\_id` VARCHAR(150) NOT NULL,

CONSTRAINT `fk\_wechatauth\_profile` FOREIGN KEY(`user\_id) references `tb\_person\_info`(`user\_id`)

)engine=InnoDB auto\_increment=1 default charset=utf8;

ALTER TABLE `tb\_wechat\_auth` ADD UNIQUE INDEX(`open\_id`);

注：CONSTRAINT `fk\_wechatauth\_profile` FOREIGN KEY(`user\_id) references `tb\_person\_info`(`user\_id`)

该表的user\_id与用户类的user\_id关联

ALTER TABLE `tb\_wechat\_auth` ADD UNIQUE INDEX(`open\_id`);追加open\_id为唯一索引，增加查询速度（相当于偏旁部首，但是过多要维护）

注意最长为767字节，意味着767/4=191为最大，`open\_id` VARCHAR(150)超过191会报错

本地账号

CREATE TABLE `tb\_local\_auth`(

`local\_auth\_id` INT(10) NOT NULL PRIMARY KEY AUTO\_INCREMENT,

`user\_id` INT(10) NOT NULL,

`create\_time` DATETIME DEFAULT NULL,

`last\_edit\_time` DATETIME DEFAULT NULL,

`user\_name` VARCHAR(128) NOT NULL,

`PASSWORD` VARCHAR(128) NOT NULL,

unique key `uk\_local\_profile`(`username`),

CONSTRAINT `fk\_localauth\_profile` FOREIGN KEY(`user\_id`) REFERENCES `tb\_person\_info`(`user\_id`)

)ENGINE=INNODB AUTO\_INCREMENT=1 DEFAULT CHARSET=utf8;

注：unique key `uk\_local\_profile`(`username`),指定uesrname唯一

头条

CREATE TABLE `tb\_head\_line`(

`line\_id` INT(100) NOT NULL PRIMARY KEY AUTO\_INCREMENT,

`line\_name` VARCHAR(1000) DEFAULT NULL,

`line\_link` VARCHAR(2000) NOT NULL,

`line\_img` VARCHAR(2000) NOT NULL,

`priority` INT(2) DEFAULT NULL,

`enable\_status` INT(2) NOT NULL DEFAULT '0',

`create\_time` DATETIME DEFAULT NULL,

`last\_edit\_time` DATETIME DEFAULT NULL

)ENGINE=INNODB AUTO\_INCREMENT=1 DEFAULT CHARSET=utf8;

店铺类别

CREATE TABLE `tb\_shop\_category`(

`shop\_category\_id` INT(100) NOT NULL PRIMARY KEY AUTO\_INCREMENT,

`shop\_category\_name` VARCHAR(1000) DEFAULT '',

`shop\_category\_desc` VARCHAR(1000) DEFAULT '',

`shop\_category\_img` VARCHAR(2000) DEFAULT NULL,

`priority` INT(2) DEFAULT NULL,

`create\_time` DATETIME DEFAULT NULL,

`last\_edit\_time` DATETIME DEFAULT NULL,

`parent\_id` INT(11) DEFAULT NULL,

CONSTRAINT `fk\_shop\_category\_self` FOREIGN KEY (`parent\_id`) REFERENCES `tb\_shop\_category`(`shop\_category\_id`)

)ENGINE=INNODB AUTO\_INCREMENT=1 DEFAULT CHARSET=utf8;

//外键连接上级id也是店铺类别id,要不为空要不为某种店铺

店铺

CREATE TABLE `tb\_shop`(

`shop\_id` INT(10) NOT NULL AUTO\_INCREMENT,

`owner\_id` INT(10) NOT NULL COMMENT '店铺创始人',

`area\_id` INT(5) DEFAULT NULL,

`shop\_category\_id` INT(11) DEFAULT NULL,

`shop\_name` VARCHAR(256) NOT NULL,

`shop\_desc` VARCHAR(1024) DEFAULT NULL,

`shop\_addr` VARCHAR(200) DEFAULT NULL,

`shop\_img` VARCHAR(1024) DEFAULT NULL,

`phone` VARCHAR(128) DEFAULT NULL,

`priority` INT(3) DEFAULT '0',

`create\_time` DATETIME DEFAULT NULL,

`last\_edit\_time` DATETIME DEFAULT NULL,

`enable\_status` INT(2) NOT NULL DEFAULT '0',

`advice` VARCHAR(255) DEFAULT NULL,

PRIMARY KEY(`shop\_id`),

CONSTRAINT `fk\_shop\_area` FOREIGN KEY (`area\_id`) REFERENCES `tb\_area`(`area\_id`),

CONSTRAINT `fk\_shop\_profile` FOREIGN KEY (`owner\_id`) REFERENCES `tb\_person\_info`(`user\_id`),

CONSTRAINT `fk\_shop\_shopcate` FOREIGN KEY (`shop\_category\_id`) REFERENCES `tb\_shop\_category`(`shop\_category\_id`)

)ENGINE=INNODB AUTO\_INCREMENT=1 DEFAULT CHARSET=utf8;

商品类别

CREATE TABLE `tb\_product\_category`(

`product\_category\_id` INT(11) NOT NULL PRIMARY KEY AUTO\_INCREMENT,

`product\_category\_name` VARCHAR(100) NOT NULL,

`priority` INT(2) DEFAULT NULL,

`create\_time` DATETIME DEFAULT NULL,

`shop\_id` INT(20) NOT NULL DEFAULT '0',

CONSTRAINT `fk\_procate\_shop` FOREIGN KEY (`shop\_id`) REFERENCES `tb\_shop`(`shop\_id`)

)ENGINE=INNODB AUTO\_INCREMENT=1 DEFAULT CHARSET=utf8;

商品

CREATE TABLE `tb\_product`(

`product\_id` INT(100) NOT NULL AUTO\_INCREMENT,

`product\_name` VARCHAR(100) NOT NULL,

`product\_desc` VARCHAR(2000) DEFAULT '',

`img\_addr` VARCHAR(2000) DEFAULT NULL,

`normal\_price` VARCHAR(100) DEFAULT NULL,

`promotion\_price` VARCHAR(100) DEFAULT NULL,

`priority` INT(2) NOT NULL DEFAULT '0',

`create\_time` DATETIME DEFAULT NULL,

`last\_edit\_time` DATETIME DEFAULT NULL,

`enable\_status` INT(2) NOT NULL DEFAULT '0',

`product\_category\_id` INT(11) DEFAULT NULL,

`shop\_id` INT(20) NOT NULL DEFAULT '0',

PRIMARY KEY(`product\_id`),

CONSTRAINT `fk\_product\_procate` FOREIGN KEY (`product\_category\_id`) REFERENCES `tb\_product\_category`(`product\_category\_id`),

CONSTRAINT `fk\_product\_shop` FOREIGN KEY (`shop\_id`) REFERENCES `tb\_shop`(`shop\_id`)

)ENGINE=INNODB AUTO\_INCREMENT=1 DEFAULT CHARSET=utf8;

商品详情图片

CREATE TABLE `tb\_product\_img`(

`product\_img\_id` INT(20) NOT NULL PRIMARY KEY AUTO\_INCREMENT,

`img\_addr` VARCHAR(2000) NOT NULL,

`img\_desc` VARCHAR(2000) DEFAULT NULL,

`priority` INT(2) DEFAULT '0',

`create\_time` DATETIME DEFAULT NULL,

`product\_id` INT(20) DEFAULT NULL,

CONSTRAINT `fk\_proimg\_product` FOREIGN KEY (`product\_id`) REFERENCES `tb\_product`(`product\_id`)

)ENGINE=INNODB AUTO\_INCREMENT=1 DEFAULT CHARSET=utf8;

4.5配置

1.引入相关jar包（配置pom.xml）

2.配置jdbc.properties(驱动器访问url地址，访问数据库，使程序与mysql通信

//驱动

**jdbc.driver**=**com.mysql.jdbc.Driver**

**// useUnicode=true支持中文  
jdbc.url**=**jdbc:mysql://localhost:3306/o2odb?useUnicode=true&characterEncoding=utf8  
jdbc.username**=**root  
jdbc.password**=**root**

1. 配置mybatis-config.xml
2. 配置spring相关

Dao,service,……..

获取读取数据库连接参数的文件位置—根据该文件配置数据库连接池—生成产生数据库连接池的对象—传入自动创建的连接池对象，告诉spring要扫描的包位置dao层（注入）

Web-inf下的文件不能通过外部url访问，保证安全性，可以通过controller分发到程序内部访问

1. 配置web.xml进行整合
2. 总结

在pom.xml指定工程依赖的jar包-----在jdbc.propeties指定数据库连接方式-----mybatis-config.xml配置mybatis----创建spring-dao.xml加载进jdbc.propeties和mybatis-config.xml，并且创建DataSource（连接池），配置mybatis与数据库的交互方式------配置spring-service.xml将创建好的DataSource注入进事务管理，便于service层操作------配置spring-web.xml,定义controller的一些行为，and将dispatcher作为一个servlet注入相应前端请求，注入spring配置文件整合。

Dispatche：拦截符合要求的请求，并把请求分配到控制器，将响应结果分给客户端

IOC：媒婆，告诉spring我们需要的类型，IOC找到符合的对象推送给我们，在spring中通过DI和依赖注入实现的，组件间的依赖关系在运行期决定，由容器动态的将某种依赖关系注入。比如对象A要操作数据库，正常是自己编写代码获得connection对象，有了spring只要告诉A需要connection，A不需要知道怎么制造，何时制造，全由spring在合适的时间制造并注入。

AOP：面向切片编程，比如操作数据库是需要权限验证，但是我们不希望将权限验证的代码杂糅在代码了，这时使用AOP思想，在增删改查时动态的将权限验证代码植入到增删改查代码中前面，实现方式为动态代理。

ORM：通过关系映射，mybatis技术，将程序中的对象自动持久化到关系数据库中

4.6测试

1.测试dao(直接操作dao层查询数据库

以打印所有Area列表为例子，在dao中实现AreaDao接口，因为mybatis的特性，不需要生成实现类，在mapper中生成对应AreaDao.xml即可完成操作

2.测试service(通过service调用dao查询数据库

接口---实现类-----注解注入

@Autowired  
**private** AreaService **areaService**;

@Autowired：有了此注解，当我们用这个接口，spring自动注入其实现类

因为实现类上标注了@service

3.测试controller(通过调用service调用dao查询数据库

//get的参数值在url上，不安全

@Controller//告诉spring，此类时controller  
@RequestMapping(**"/superadmin"**)//设置调用的路径  
**public class** AreaController {  
}

**package** cn.xww.o2o.controller.superadmin;  
  
**import** cn.xww.o2o.domain.Area;  
**import** cn.xww.o2o.service.AreaService;  
**import** org.springframework.beans.factory.annotation.Autowired;  
**import** org.springframework.stereotype.Controller;  
**import** org.springframework.web.bind.annotation.RequestMapping;  
**import** org.springframework.web.bind.annotation.RequestMethod;  
**import** org.springframework.web.bind.annotation.ResponseBody;  
  
**import** java.util.ArrayList;  
**import** java.util.HashMap;  
**import** java.util.List;  
**import** java.util.Map;  
  
@Controller  
@RequestMapping(**"/superadmin"**)  
**public class** AreaController {  
 @Autowired  
 **private** AreaService **areaService**;  
 @RequestMapping(value = **"/listarea"**,method = RequestMethod.***GET***)  
 @ResponseBody//以json形式返回给前端  
 **private** Map<String,Object> listArea(){  
 Map<String,Object> modelMap = **new** HashMap<String, Object>();  
 *//获取Service层返回的区域列表* List<Area> list = **new** ArrayList<Area>();  
  
 **try** {  
 list = **areaService**.getAreaList();  
 modelMap.put(**"rows"**,list);  
 modelMap.put(**"total"**,list.size());  
 }**catch** (Exception e){  
 e.printStackTrace();  
 modelMap.put(**"success"**,**false**);  
 modelMap.put(**"errMsg"**,e.toString());  
 }  
 **return** modelMap;  
 }  
}

5.logback(日志组件)日志配置



Logback:标签

Logger:日志类型

Appender:日志输出的目的地

Layout:格式化

日志输出：在tomcat启动的CATALINA\_BASE: "C:\Users\25293\.IntelliJIdea2017.3\system\tomcat\Unnamed\_o2o"目录下(配置的)，有日志文件，记录了时间-所属线程-属于哪个类-相应日志信息

1. 开发

一：店家管理

6.1优先开发店家管理系统

a.店铺注册

连接数据库

Mybatis数据库变映射关系的配置

Dao🡪service🡪controller层代码编写，Junit的使用

Session，图片处理工具Thumbnailator的使用

*<!--图片处理-->  
<!-- https://mvnrepository.com/artifact/net.coobird/thumbnailator -->*<**dependency**>  
 <**groupId**>net.coobird</**groupId**>  
 <**artifactId**>thumbnailator</**artifactId**>  
 <**version**>0.4.8</**version**>  
</**dependency**>

Suimobile前端设计与开发

添加店铺的返回类型（dto）查看添加状态给controller处理

Service层（加事务，任何一步错误需要事务回滚）

将店铺信息插入到数据库🡪返回店铺id🡪根据id创建存储图片的文件夹🡪在该文件夹下处理图片🡪将文件夹地址更新回程序

Controller层

从前端获取request信息，解析后传给控制器

6.2前端设计

利用工具SUI Mobile（自适应网页设计）



6.3使用前后端交互，html页面读取数据库信息，html写入的信息存储到数据库

Js:

商品分类和区域从后台获取，填充到前端控件中

点击提交网页表单的内容全部获取到通过Ajax发给后端

验证码设计

1. 在pom.xml中引入jar包
2. 在web.xml中定义servlet
3. 在shopoperation.html中生成验证码控件，利用changeVerifyCode方法调用servlet生成验证码
4. 在js中获取到验证码，并且传入给后台判断
5. 在后台controller生成验证码比对

6.5外传拓展：Mysql主从分离实现

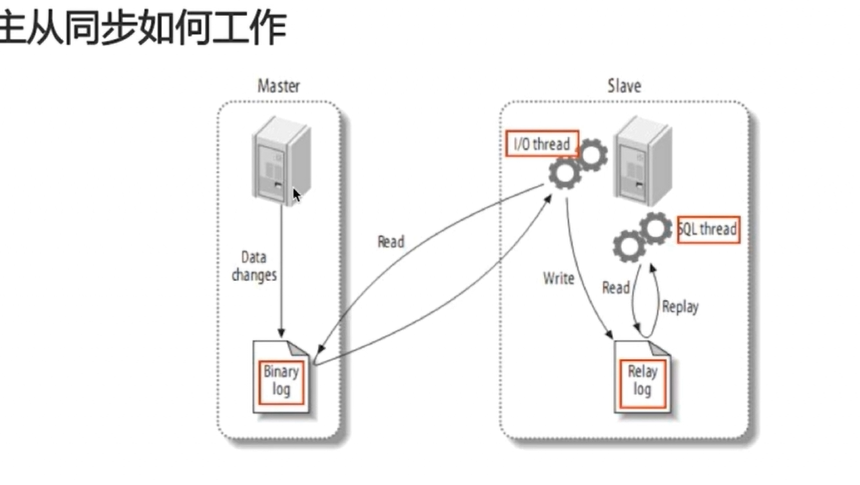
1.实现读写分离(主库写，从库读)2.将数据同步到好几台数据库上，减轻同时访问某一台数据库读写压力

学习目标：

数据库层面的主从配置实现

代码层面的读写分离实现

a.主从同步实现



1. 主服务器Master将数据变换记录到二进制日志B..log
2. 从服务器slave通过IO线程读取B..log同步到自己的日志R..log
3. Slave通过SQL线程插入与主服务器一样的数据操作

设计原因：B..log不能一下传过来，通过new两个线程，可以在IOthread的同时进行sqlthread(异步),并且此时slave还可以去做其他事，比B..log一下传过来再操作要快

b.服务器实际操作配置(要两台服务器，或者下两个mysql设置两个端口)

c.代码层读写分离（主库数据源用来写，从库的用来读）

二：店铺信息

店铺信息编辑：实现单个店铺信息获取和编辑更改

1. dao层（根据id查询店铺信息，modify店铺信息）
2. service层

3.controller层（接受前端修改信息）

4.前端设计

补Session:在web中服务器为浏览器创建一个会话对象session，在需要保存用户信息时，服务器将用户信息写到浏览器独占的session中，当浏览器访问服务器时，服务器从session中取出用户数据为用户服务。服务器创建session后会以cookie的形式回写给客户机，只要客户端浏览器不关，都会带着session的id去访问服务器，服务器给出与该session对应的服务，tomocat默认seesion过时时间为30min。一般在你登录时会将一个user信息传给服务器，服务器创建对应session

三：实现商店列表(查询owner创建的商店,分页查询店铺

LIMIT #{rowIndex},#{pageSize};//从rowindex行取返回pagesize

行数据

Dao—service—controller

前端开发（商店列表，suiMobile的栅格）

四：商品类别信息相关模块开发

1.根据shopid查询旗下所有商品类别

Dao—service—controller—前端

类别管理

2.批量添加商品

3.商品类别删除

五．商品相关功能(product相关

1.商品添加

Dao—service—controller—前端

2.商品编辑

Dao—service—controller—前端

3.商品列表管理

Dao—service—controller—前端

*//dao只认函数，浏览器认页数，将pageindex转成rowindex***public class** PageCalculator {  
 **public static int** calculateRowIndex(**int** pageIndex, **int** pageSize) {//从第几页开始查pagesize  
 *//数据库中分页语句 LIMIT #{rowIndex},#{pageSize};从rowIndex，查pageSize条数据  
 //如果pageindex=1，则从第0条开始选取pagesize条，pageindex=2,表示要查第二页，2-1=1，从第pagesize开始选取了，。。。=3，从第2\*pagesize选* **return** (pageIndex > 0) ? (pageIndex - 1) \* pageSize : 0;  
 }  
}

六．前端展示系统的开发

向用户展示店铺与商品信息

1．首页开发

头条读取以及滚动播放(headline

一级类别列表获取(没有parentid

Service层貌似只是做转发没什么用，其实不是，在缓存时service能让dao和controller解耦

2.店铺列表页开发

分页读取以及无极滚动

多条件排列组合查询店铺信息

3.店铺详情页开发

与2差不多

4.总结

七．项目部署

阿里云部署Java网站和微信开发调试心得技巧(上)\_慕课手记 <http://www.imooc.com/article/20583>

http://www.imooc.com/article/20584

阿里云部署

微信开发调试

1. 申请阿里云服务器
2. 搭建出程序执行环境
3. 在服务器上发布并允许自己的web project
4. 域名解析

申请的是Windows服务器，申请Linux较好，就是在服务器上安装运行的环境，将本机项目打包成war放到服务器tomcat的webapp下就可以了

1. 微信测试号的申请与连接以获取微信用户信息

WechatUtil，这个类里有注册的id和url，根据自己注册的填写

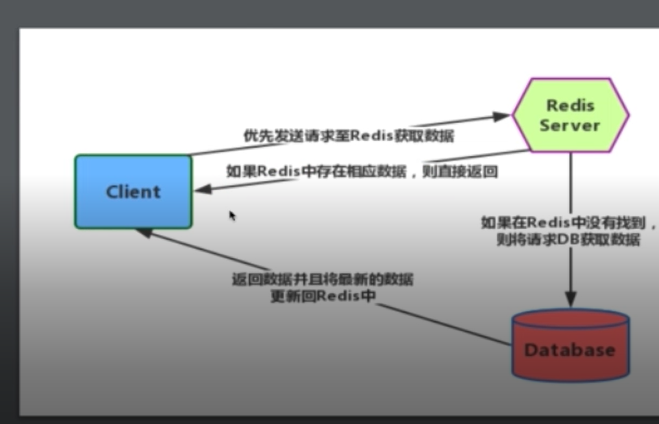
根据WechatLoginController里面的路由，根据此路由访问微信定义好的url,url里带有appid和redirect\_uri,将需要用的参数code等传递到controller，获得code的时候获取token,token里带有微信用户信息

八．缓存优化

加密工具：DESUtil

1.什么是redis和jedis

一个key-value的存储系统，value支持各种值，redis操作是原子性（事件要不成功，要不回滚），redis会定期将数据写入数据文件（rdd模式），优点是单独子线程写入，主线程不会有io操作，很高效。缺点是隔断时间持久化,中途redis损坏会丢失数据。



将不常更改的数据存入redis（头条信息，店铺类别，区域列表等）

Jedis:服务于java的redis客户端

2.redis配置

引入jar包—配置springredis.xml—改造dao,service实践类，引入redis

1. jedis基本通用函数配置及使用

九．平台账号登录功能

这里我做了简单处理，当用户名和密码用数据库中信息匹配，登录…./admin/login会进入店铺展示界面，当访问…/admin/login?userType=back时，将进入店铺管理。输入用户名和密码错误超过三次要做验证码验证，防止机器刷密码。