

1. 项目名称 tmall_springboot

2. java源代码包结构

pojo 实体

config 配置

es elasticsearch dao类

exception 异常处理

realm shiro相关

dao DAO类

interceptor 拦截器

web 控制层

service Service层

test 测试类

util 工具类

comparator 比较类

3. webapp 目录

css css文件

img 图片资源

js js文件

4. templates

thymeleaf 模板文件

天猫整站 -springboot

- redis：数据库缓存，nosql（图形界面）
- nginx：反向代理，动静分离 → 集群的负载均衡 LVS
- elasticsearch：搜索引擎（kibana 工具，数据分词）
- shiro：登录验证的安全框架
- restful 标准：
 - ① 资源名称用复数，而非单数

② CRUD 对应
△
增加：post
删除：delete
修改：put
查询：get

③ id 参数的传递用 /id 方式

④ 其他参数采用 ?name=value 形式

⑤ 返回数据

查询多个返回 json 数组

增加、查询一个，修改，返回当前 json

删除 返回空

前后端分离：

1、html 页面看成包含数据和不包含数据两部分。

先准备一个不包含数据的，传给浏览器，很快

再通过 Ajax 技术，仅仅从服务器异步获取数据

小：用户体验比较好，先看到页面，根据加载中……
后端只提供数据，前后端开发降低了耦合度，开发效率提高。

2. 前端：AJAX .JSON 数据传输格式 key:value, []

 原生Ajax：异步刷新

jQuery对ajax的支持：\$.ajax({url,data,success})

 \$.get(page, {name:value},

 function() {})

 \$.post 同\$.get参数

\$("#id").load(page, [data]) load

 用于显示ajax服务端文本的元素id → 服务端页面

JavaScript对象分为内置对象 Number, String, Array, Date

 自定义对象 JSON

string → json eval("()")

json → string stringify(xx)

3. 前后交互：J2EE, SSM, spring boot

 servlet

 springMVC

 → vue.js + restful + pagehelper + thymeleaf + sboot

风格

分页

纯html

33

Shiro: 宽泛的安全框架，主要针对用户验证和授权操作。

1. RBAC: Role-Based Access Control 基于角色的访问
eg 你要能删除产品，那么当前用户有~~产品~~经理角色
Resource-Based Access Control 基于资源的访问
eg 你要能删除产品，那么当前用户有删除产品权限
e.g. 用户 - 角色 - 权限

用户：角色 = n:n

用户：权限 = n:n

Realm: 当应用程序向Shiro提供了账号和密码后，Shiro就会问Realm这个账号密码对不对，若对，该用户拥有哪些角色，哪些权限

实际上，Realm 就是中介，得到了Shiro给的用户和密码后，像从shiro.ini，也可以从数据库中去查询
这个是真正进行用户认证和授权的关键

DatabaseRealm: 通过数据库验证用户和相关授权的类。extends AuthorizingRealm

doGetAuthenticationInfo() 验证

doGetAuthorizationInfo() 授权

2. 加密

▷ md5 加密算法，Md5Hash()，非对称的

缺陷：比如123的md5都是一样的，那通过反推就破了。

：引用了盐的概念，炒菜，直接用md5，就是对食材（源密码）进行炒菜，因为食材一样，炒出来的味道一样，可是如果加了不同的盐分，即便食材一样，味道不一样。

123+随机数 → 保存在数据库中。

加密次数也是不同的，所形成的密文不一样的。

▷ 对数据库调整，将盐巴保存，否则无法判断密码的一致性。

3. Web 支持 (servlet), SSM, 权限维护一套。

url 配置权限，springboot

Redis：如同 HashMap，取东西很快，不在 JVM 中运行，而是以一个独立进程的形式运行。

是一个开源的使用 ANSI C 语言编写、支持网络、可基于内存亦可持久化的日志型、key-value 数据库，并提供多种语言的 API。

一般会被当成缓存使用，比 mysql 快，提高了性能。

1. nosql 非关系型数据库

运行第一次从数据库中取，随后存放在缓存中

key-value

⇒ 二进制 → 转化为 json 格式 redis 配置

2. Springboot 默认支持缓存

@EnableCaching

3. 在 服务层 做缓存

@CacheConfig 分表示在缓存里的 keys

@Cacheable(key = "xx-one-' + #P0") 获取一个 对象

@Cacheable(key = "xx-page-' + '#P0 + '-' + #P1") 分页集合

@CacheEvict(allEntries = true)

增加、删除、修改 key 被删除

若采用 @CachePut(key = "xx-one-' + #P0"), 可以在 redis 中增加一条数据, 但它不能更新分页缓存里的数据
这就会出现 数据不一致 的问题

非常规 ⇒ 精细做法, 同时更新分页缓存里的数据, 但它是 'nosql', 会超级复杂, 所以折中方法是, 一旦增加某个分类数据, 就清除掉缓存中的所有, 这样, 牺牲了一点小小的性能, 但保障了数据的一致性

没有直接调用 `listByCategory()`

是因为 AOP 编程，在 Spring Boot 中若这个方法一开始就没缓存，那么往涉及到的方法也不会走缓存。所以为了防止出现这种情况，调用自己，走缓存，从而被 AOP 切面编程所拦截到。

版本问题：

Elasticsearch 版本高，Spring Boot 也提供了对其中的 JPA，叫 Elasticsearch Repository，但是 JPA 没跟上时代，即 JPA 不支持高版本的 ES，只能用低版本 ←

④ Spring Boot 的缓存机制是通过切面编程 AOP 实现。

因为在 ProductService 的 `fill()` 中调用了 `listByCategory()`，
AOP 拦截不到，也就不会走缓存，而还是从数据库中去取，
所以统一的方式诱发 AOP，才会走 Redis 缓存。

→ 提供了一个工具 `extends ApplicationContextAware`，其中 `inSetApplicationContext` 方法，致使 Spring Boot Application 运行启动时，会将其注入 `applicationContext`，调用 `getRea-`

ElasticSearch：Lucene，Solr 常见的 Java 搜索引擎

- Lucene：这个开源项目，使 Java 开发人员很方便地得到像搜索引擎 Google, Baidu 那样的效果

- Solr：以连接数据库为类比，Lucene → JDBC，基本 8983 Solr → mybatis，方便访问、配置和调用，可视化配置界面

- Elasticsearch：基于 Lucene 封装，一个可视化工具 客户端 9200
工具 Kibana (5601)

1. 将 JPA 的包和 Elasticsearch 的包分开放，会有冲突异常
同时在 application 启动类指定包名。
→ JpaInDAO 做了连接 Redis 的，放一起会有影响

spring boot 使用 elasticsearch 到 jpa, or

ElasticSearch Repository

2. 查询前先初始化, 因为 elasticSearch 中没有数据
先从数据库中查找, 然后放到 es 中;

nginx: 反向代理和动静分离, 只需配置工作即可完成

① 采用 80 端口, 反向代理到 9090. 这样用户访问, 就

用 80;

proxy-pass http://11127.0.0.1:9090

② 动静分离, 对于图片、css、js 等一系列静态资源, 直接走 nginx, 而不再通过 tomcat

location ~ \.(css|js|png|ttf|woff2|eot|svg|map|jpg|gif)\$ {

root tomcat 7.0\webapps\路径 1.1

即使服务器崩掉, 浏览器依旧还能看原打开的图片

start nginx 启动

nginx -s stop 关闭

技术准备：

Java：基础和中级

前端：html, CSS, Javascript, JSON, AJAX, JQuery,
Bootstrap, vue.js

框架：spring, springmvc, springboot

中间件：redis, nginx, elasticsearch, shiro

数据库：MySQL

开发：IDEA, Maven \Rightarrow Spring Initializer

开发流程：

需求分析：前台、后台

表结构设计：表与表之间的关系，建表SQL语句

原型：与客户沟通顺畅的项目设计流程里要有

后台一分类管理：查询、分类、增加、删除、编辑、修改

后台其他管理：属性、产品、属性值、产品图片、用户订单

前台 - 首页

前台无需登录：访问页

前台需要登录：购物流程

• idea2017.2.7 版本，

导入 maven 项目后，pom 中依赖及 plugin 报错

解决：File → maven → Always update snapshots

另外：view → tool windows 里有多种可显示的工具栏
maven projects 里，先 clean，再 install，最后
点击左上角的 刷新

即可正常

• 下载 maven 3.5.0，本地仓库，并配置到 path 中

修改了两个配置 ① 从 阿里云 下载 <mirror>

② 修改 仓库位置 d:\jsphj\maven\...

JPA: Java 持久化规范

java persistence api

CRUD + 分页 JpaRepository

新增 + 修改 save()

由实体类 id 是否为 0 决定

查询:

① 参数: 当前第几页 start, 默认 0
每页显示数据 size, 默认 5

② start 为负, 修改为 0

当前是首页, 还点击上一页

③ 设置倒排序

这个 page 中, 包含分页信息和 sort (Sort.Direction.DESC, "id")

根据信息

(getContent)

page.number 当前页

page.totalPages 总页数

区别:

	J2EE	SSH	SSM	Springboot
前台功能	√	√	√	√
后台功能	√	√	√	禁用了 hibernate JPA
存储层	JDBC	Hibernate	MyBatis	
控制层	Servlet + 反射	Struts	Spring MVC	Spring MVC
IDE	Eclipse	Eclipse	IDEA	IDEA
是否 maven	否	否	是	是
项目格式	标准 Java	动态 Web	Maven	Maven
启动方式	Tomcat 独立	Eclipse 中的	IDEA 中的	Spring Boot 自带
模板技术	JSP	JSP	JSP	Thymeleaf 并行
前框架	Bootstrap	Bootstrap	Bootstrap	不用服务端直接
前后端分离	✗	✗	✗	✓
异步处理	JQuery	JQuery	JQuery	axios.js
RESTFUL	✗	✗	✗	标准风格
动静分离	✗	✗	✗	nginx
安全框架	✗	✗	✗	shiro
缓存	✗	✗	✗	redis
		✗	✗	elasticsearch

Page<?> page = dao.findAll(pageable)

需求分析-展示

首页、产品页、分类页、搜索结果页、购物车查看页、结算确认支付页、支付成功页、我的订单页、确认收货页、评价页、页头信息展示

需求分析-交互

通过POST、GET等http协议，与服务端进行同步或异步数据交互。

分类排序、立即购买、加入购物车、调整订单项数量、删除订单项、生成订单、订单页功能、确认付款、确认收货、提交评价信息、登录、注册、退出、搜索

需求分析-后台

分类、属性、产品、产品图片、产品属性、用户管理、订单

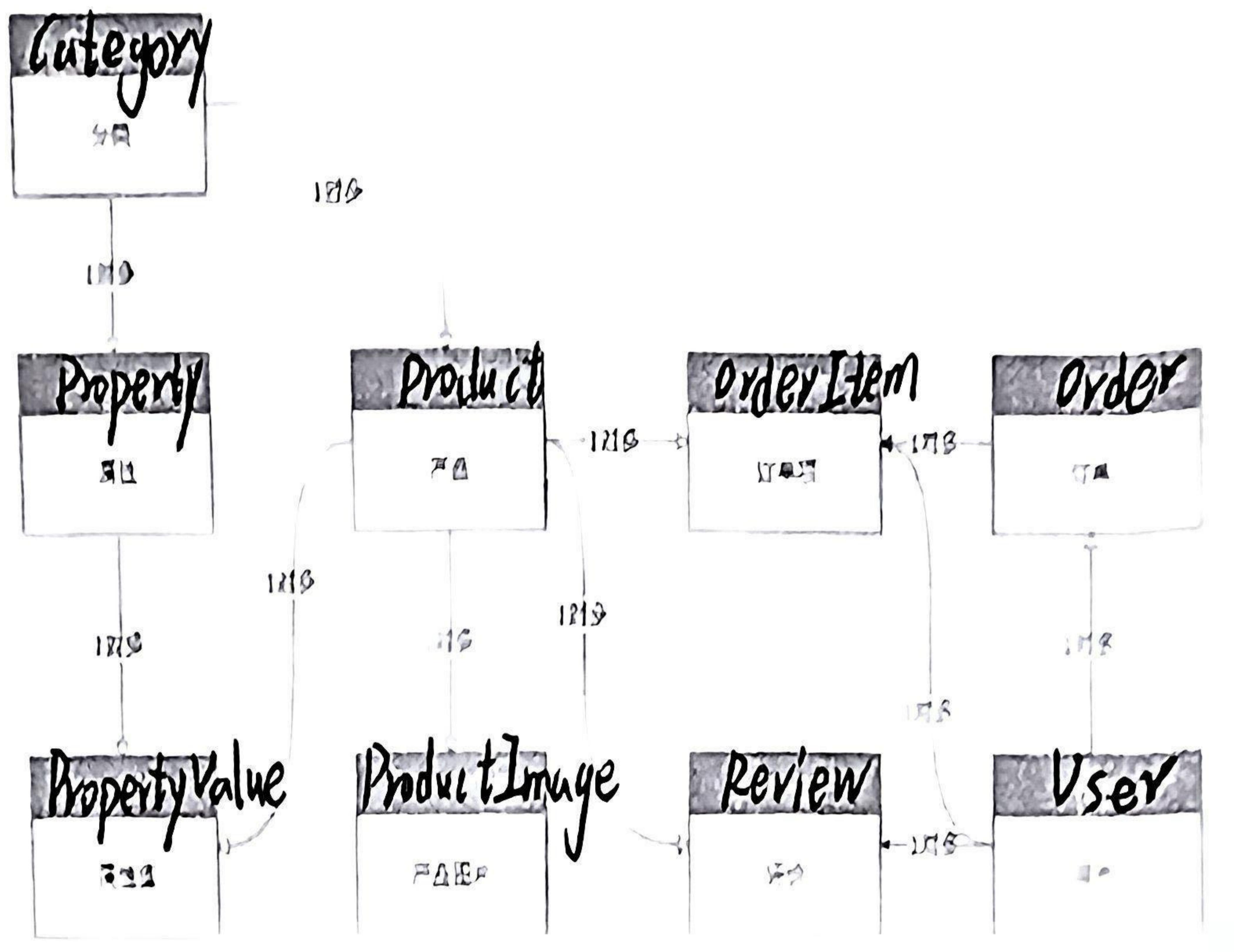
表结构设计 同前 ssh 项目

原型-后台

后台-分类管理

1. 静态资源：图片 img, 样式 css, 脚本 js
moment: 日期格式
jquery
vue
bootstrap
axios: ajax 库

2. html 包含关系 <template th:fragment="html" >



3. 查询

1>category: @Entity 一个实体类

@Table (name = "category") 对应表名

@JsonIgnoreProperties ("handler", "handlerLazy...")

因为要做前后端分离，而前端端根据交互用的

是json格式，那后端Category对象会被转换为JSON
数据，而本项目使用jpa做实体类持久化，
jpa会默认使用hibernate，在jpa工作过程中，会
创造代理类来继承Category，并添加handler

hibernateLazyInitializer 这两个无须json化的属性，
所以用Ignore... 来忽略掉

2>categoryDuo: 集成了Jpa Repository，可分页、CRUD

3>categoryService: @Service 标注 Service类

@Autowired 自动装配 Duo

Sort sort = new Sort (Sort.Direction.DESC, "id")

4>AdminPageController: @Controller 一个控制器

@GetMapping ("")

redirect: xx 客户端跳转

"admin/xx" 访问 xx.html 文件

5>CategoryController: 专门用来提供 RESTFUL 服务

@RestController: 方法的返回值自动转为JSON
数据格式

@Autowired: 自动装配到 service

@GetMapping: Categorys 对象集合，会自动转换为 JSON 的数组
劫持浏览器

6) Application 启动类 @SpringbootApplication

7) CORSController: 配置类，用于允许所有请求都跨域
※二次请求，第一次是获取html页面，第二次通过html
面上的js代码异步获取数据，一旦部署到服务器就
容易面临跨域请求问题，所以允许所有访问都跨
域，就不会出现通过 ajax 获取数据获取不到的问题。

8) Global Exception Handler 异常处理

主要是在删除父类信息的时候，因为有外键约束存在，而
导致违反约束。

9) application.properties:

- ① 数据库访问地址、账号密码、驱动以及表结构自动生
成策略。
- ② thymeleaf，springboot 的官方推荐视图，纯 html
- ③ 上下文地址

④不要缓存

⑤设置上传文件大小

⑥ jpa命名：驼峰

10) ListCategory.html 获取数据 + 展示数据

① 获取 \$(function(){}); jquery 表 html 加载好后执行

• var data4Vue = { uri: me中用到的数据
 beans: [] }

• var vue = new Vue({ el: '#workingArea',

跟本页面 <div id=> data = data4Vue,
对应 }) 创建 Vue 对象 上面定义的

• mounted: function() { this.list(); }

加载 vue 对象成功之后会调用

• methods: { list: function() {

var url = this.uri;

axios.get(url).then(function(response){

vue.beans = response.data;

});

}

调用 axios.js 这个 ajax 库，进行异步调用，成功后，把服务端返回的数据，保存在 vue.beans

vue.js: vue出来的一个脚手架，组件式地开发

node.js: javascript一个浏览器运行的脚本语言，控制
前端元素，即纯粹的客户端语言

要与服务端交互，就得等服务端开发……

——偏多以java为主。

有人想，要服务端也用javascript开发，那服务端
也不容易开发服务端。

→大佬开发了一个v8引擎，在服务端运行
javascript语言，有一定的发展，便出现了可在
服务端运行的javascript，叫node.js。

服务端运行的javascript，叫node.js。

webpack: 包含前端的项目，多个.js，多个.css，多个静态图片。

多个其他前端资源

webpack 将多个静态资源打包成一个叫 assets 的

②展示数据 <v-for="bean in beans">

<td>\$ {bean.id}</td>

⇒ vue.js → 没有用是因为适合做单页面开发，本项目是
复杂的多页面。

4. 分页，Page 和 Navigator，包含 jpa 提供的分页类，除了

<el<1 5 6 7 8 9 10 11 >>

前台显示3个，共7个分页点
我们在构造方法里加了一个参数，并封装了 Page 类

⇒ CategoryService，增加 list() 方法

⇒ CategoryController 修改 list()

参数中有 @RequestParam 表示

(name, value, defaultValue, required)

将请求参数区根据映射到功能处理方法的参数

3) list(category.html): list() 带上参数

包含了 adminPage.html

5. 增加

⇒ listCategory.html 页面中含有 add

• var data4Vue = { add bean 和 file 属性

• getFile: function(event) {

this. file = event.target.files[0]

3. 调用 axios 的 post 方法进行实际工作

4. 上传成功，调用 vue.listto 重新查询第一页数据，然后还原，使得输入部分回到上传前的状态
(有步骤分离相对麻烦)

add 1. 判断是否为空 (分类名、分类图片)
2. 采用 axios 上传图片 得用 formData 这种方式
var formData = new FormData();
formData.append("image", this.file);
formData.append("name", this.bean.name);
axios post url, formData).then() { }

27 add inTable

37 categoryService.add()

47 ImageUtil 工具类 ① change2jpg . 保留图片文件的进度
格式为 jpg

② resizeImage 改变图片大小

57 categoryController.add()

① 通过 categoryService 保存到数据库

② 接受上传图片，保存到 img/category 目录下

③ 文件名用新增分类的 id

④ 目录不存在，则创建

⑤ image.transferTo 进行文件复制

⑥ BufferedImage img = ImageUtil.change2jpg

⑦ ImageIO.write(img, "jpg"); file);

6 删除

1) listCategory.html: delete Bean function (id) {

delete
保留参数

· checkDeleteLink()

· 组织url 规范是 /category/123

· axios.delete

· 根据REST规范，删除后，应该回

一个空字符串，所以会判断为

0 == response.data.length

这个断句，在 删除 按钮时点击。@click="deleteBean(id)"

2) CategoryService: delete()

3) CategoryController: delete() DeleteMapping("xx/{id}")

① 由 id 删 根据库中的数据

② 删除对应文件。file.delete()

③ 返回 null，会被 REST Controller 转换为空字符串

7. 编辑

1) AdminPageController: 加映射，value = '/edit'

return "admin/edit(category)"

2) editCategory.html: 调用 get()

- ①先从地址栏获取 id `getDrlParams("id")`
- ②组织 url 为 /xx/m/ 格式
- ③ `axios.get(url).then()`
- ④ 把返回对象放在 bean 上
- ⑤ vue 根据 `v-model` 自动把 bean 上数据放入框中

3) categoryservice

4) categoryController `@PathVariable` 获取 url 的变量

8 修改 `> update()`

类同 `add`, 注意图片文件的上传用 `formdata`

2) categoryService

3) categoryController

① 获取参数名称，用 `request.getParameter("name")`

② `categoryService.update()`

③ 若上传了图片，用 `add` 时共用的方法

④ 返回对象

后台 - 其他管理

1. 属性管理

1) `property: @ManyToOne`

外键约束

`@JoinColumn(name = "cid")`

orderService 中的一个奇怪方法 removeOrderFormOrderItem，作用是把订单里的订单项的订单属性设置为空。

有个 Order，拿到它的 orderItems，把 orderItems 的 order 属性设置为空。

为什么要这样？

SpringMVC 的 RESTFUL 注解，在把一个 order 转换为 json 的同时，会把其对应的 orderItems 转换为 json 对象。而 orderItems 对象上有 order 属性，这个 order 属性又会被转换为 json 对象，然后这个 order 下又有 orderItems ……，产生无穷递归，系统报错。

那为何不用 @JsonIgnoreProperties，因为后端用到了 redis，有 bug，不能用。

PropertyDuo：CRUD + 根据分类查询 接口

PropertyService：业务上查询分类下的属性，所以 list() 会带上分类的 id

PropertyController：不同路径的访问

listProperty.html + editProperty.html

2. 产品管理 跟属性管理几乎一样

3. 产品图片管理 { 详情图片
单个图片 }

产品中 firstProductImages，在 list 方法中，set XX 为 page.getContent()，并在页面显示出来

4. 产品属性值 修改、查询、初始化

1. 由产品得分类，获取分类下的所有属性集方

2. 由属性id、产品id去查询，是否存在属性值

5. 用户管理 涉及到后面的匿名，用 ** 表示

只是查看，增加前名处理，注册

6. 订单管理

状态，创建后处于待发货，后台点击发货，变为待收货。前台确认收货，后台变为待评价，前台点击评价，后台变为完成

→ 点击按钮后 使用 \$(e.target).hide() 隐藏

原型前端

绿色表示已实现的需求

红色表示与服务器交互的需求

1. 首页

- 在横向导航栏上提供4个分类连接
- 在纵向导航栏上提供全部17个分类连接
- 当鼠标移动到某一个纵向分类连接的时候，显示这个分类下的推荐商品
- 按照每种分类，显示5个商品的方式显示所有17种分类

5. 产品页

- 显示分辨率均为950x100的当前商品对应的分类图片
- 显示本商品的5个差评图片
- 商品的基本信息，如标题，小标题，价格，销量，评价数量，库存等
- 商品详情
- 评价信息
- 该商品详细图片
- ✓立即购买
- ✓加入购物车

9. 分类页

- 显示分辨率均为950x100的当前分类图片
- 显示本分类下的所有产品
- ✓分类页排序

4. 搜索结果页

- 显示满足查询条件的商品

2. 购物车查看页

- 在购物车中显示订单项
- ✓更新订单数据
- ✓删除订单项

4. 结算页

- 在结算页面显示被选中的订单项
- ✓生成订单

3. 确认支付页

- 确认支付页面显示本次订单的金额总数
- ✓确认付款

3. 支付成功页

- 付款成功时，显示本次付款金额

2. 我的订单页

- 显示所有订单，以及对应的订单项

2. 确认收货页

- 显示订单项内容
- 显示订单信息，收货人地址等
- ✓确认收货

4. 评价页

- 显示要评价的商品信息，商品当前的总评价数
- 评价成功后，显示当前商品所有的评价信息
- ✓提交评价信息

4. 页头信息展示

- 未登录状态
- 已登录状态
- ✓登录
- ✓注销
- ✓退出

6. 所有页面

✓搜索 **search, simplesearch**

home.jsp

→ top.html

→ search.html

→ main.html

categoryAndCarousel.html

categoryMenu.html
productsAside
categorys.html
carousel.html

5

第18章

前18章-无需登录：注册、登录、退出、产品页、模块登录

购物页面行为

监听：立即购买按钮 + 加入购物车按钮

分类页

比较器 sort

前18章-需要登录：立即购买、结算页面、加入购物车、查看购物

登录状态的拦截 (LoginInterceptor, 配置使用)

其他拦截 (天猫首页 contextPath)

分类下 XIXIX
XIXIX

购物车文件 cartTotalNumber

购物车页面、生成订单、我的订单项、-操作

+、-商品数

付款

删除商品

确认收货

选中到结算页

评价

缓存和Shiro

1. user表，进行带盐加密 salt

2. JPARealm extends AuthorizingRealm

SimpleAuthorizationInfo

SimpleAuthenticationInfo

3. shiro配置，bean都是些常规配置

- getJPARealm() 指定了 Realm 使用 JPA Realm
- hashedCredentialsMatcher() 指定了 加密算法

4. 注册 修改为 REST Controller 的 register()

- string salt = new SecureRandomNumberGenerator().nextBytes().toString()
// 随机创建盐
- int times = 2 // 2 次加密
- string algorithmNames = "md5" // 用 "md5" 加密算法
- string encodedPassword = new SimpleHash(algorithmNames, pa, salt, times).toString()
// 最终密码

5. 登陆，通过 shiro 校验

Subject subject = SecurityUtils.getSubject();

UsernamePasswordToken token = new UsernamePasswordToken
(name,getPass...)

subject.login(token)

6. 退出，用 subject.logout()

7. 拦截器。用 subject.isAuthenticated()

Redis 6379

1. 配置，作用是使得保存在 redis 里的 key 和 value 转换为具有可读性的字符串，否则会乱码，不易观察

RedisConfig extends CachingConfigurerSupport

2 PortUtil 提醒启动工具类

3 Application @EnableCaching 启动缓存

4 CategoryService 的缓存配置 之前

其中，其他的后台管理都一样，但 productService，
productImageService, orderService 等还是有点区别
采用统一的方式故意诱发aop，走缓存

ElasticSearch 9300 基于MySQL的

将天猫整站的模糊搜索基于 elasticSearch 做

1. 下载客户端 Kibana 560

2. ProductESDao extends ElasticsearchRepository<, >

3. Application 为 es 和 jpa 指定包

4. ProductService ↳ 增加、删除、修改时，通过 ESDao 同步到

2> 初始化数据到 es initDatabase2ES

3> 查询，以前模糊查询，现通过 ESDao

到 elasticSearch 中进行查

5. Product @Document(indexName = "tmall-springboot", type = "prod")

告诉 es 如何匹配数据库

表

Nginx

动静分离、双向代理

53

Question:

1. 属性管理页面没显示

listProperty 中，list() 中的 vue.Plugin = response.data
editProperty 中，list() 中的 vue.category = this.bean.category

⇒ 面包屑导航

href = "'admin-property-list?cid=' + category.id"

2. 在分类管理中，更新的 putMapping 中带有 id，则 js 中 axios.put 中 url 也要带有 id

在属性管理、产品管理中的更新 putMapping 不带有 id，所以 axios.put 中也不要带有 id

3. 图片信息未显示

id、31号、单引号

4. 无法上传产品图片

Miss url template variable "pid" for method parameter
of type String

注解 controller 上的实参与方法里的形参名不一样

add (@RequestParam("pid") int pid)

△ 获取请求参数

5. 端详详细中，图片未显示，提示 404，找不到

路径不对，在 controller 中，根路径写错了。

img /，导致在 html 中找不到文件夹

6 后来产品页也显示不出来

page. getcontent()

7. 产品管理 → 设置属性，不出现页面

- Edit 项的数据 product，

- service 的 list 方法，PropertyValue 提示为空

- getProduct 的参数 pid，没得到产品就得靠属性

- 面包屑导航

8. 用户界面没显示东西

- data 显示数据

- 添加分页 in html

9 点击发货要刷新才能看见发货时间

- 在 js 中的 deliveryOrder() 中，刷新整个页面

或者在 `cd` 处写 没写出来

10 删了 remove 这个报错，提示写不进去.json

哭崩，这个奇怪的方法

11 在业务需一个订单数据产生的两行 tr，此时就只能在 <tr> 上进
查看详情

行 `v-for`，而是需要 `template` 标签

12. 加了引号，访问 500

. v-for=""，东西没放进去

· lspan

13. 商品分类名 {{c.name}}

· 实际取到，但页面显示空，所以就是 html 的问题

<script> 在 <head></head> ~~ 和 <body></body> 中

14. pom.xml 中的配置。

Spring boot web . tomcat 支持 热部署、jpa, redis

test, thymeleaf, elasticSearch,

缺少 jna，不然报错

thymeleaf legacyhtml5 模式支持, JPA 支持, mysql, junit, shiro, hsqldb,

插件 spring-boot-maven-plugin

⑮. 底页面分类的推荐产品列表显示不出来 ???

取到的但页面不显示，也没报错

回家后竟然可以了，想知道它经历了什么。~~~

16. 编码访问路径的问题

方法名大写了， ajax 方法中 url 忘加 pid

17. 分类页不出来，说没定义 categorys

莫名其 users，把模块登录移加进去了，在主 html 中

18. 搜索用的是post

站长说 get 中文有问题，但我 好像还行

19. 拦截器拦的主页没了

prehandler：执行 controller 方法前执行 true, false
goon response

posthandler：生成视图前执行 后处理回调方法

aftercompletion：请求处理完毕回调，仅调用处理器执行

20. 删除后刷新问题：

o == response.data.code

vue.load 更新 oif 变量

21. 提交订单，提示 oid 未定义

在 jquery 中的请求没定义 oid => 在 axios 中则用了
data4vue vue.oid XX

22. 确认支付后，商品的库存应该减少
有待完善

23. 订购完了，竟然可以刷好评，显然不允许这样。

改了没效果，用 status 判断。finish 则不显示提交
反之显示并提交

24. 注册页面，死活点了按钮没反应，因为 坑死。

el 忘记井，而 div 中的 id 多了井

25. 生成的盐，怪怪的。

随机创建盐，字符串

26 加了 redis, shiro, nginx, elasticsearch 后，点击购物车没反应，按钮不起作用了

连接首页的连接改了一个 ??? 忘了怎么改的了

27. 评估这里，同一个方法有 2 次对不同表的 dao 操作，若少选错，需要回滚，所以得事务管理

28 缓存刚开始没起作用，失效的，虽然库里有数据，但就一直从库里取，一遍遍地跑，都删除才成功

经过这个项目，我们都完成了如下的一系列典型场景功能

1. 购物车

立即购买 加入购物车 查看购物车页面 购物车页面操作

2. 订单状态流转

生成订单 确认支付 后台发货 确认收货 评价

3. CRUD

后台各种功能

8. 搜索查询 - 基于 elastic search

搜索 模糊查询

4. 分页

后台各种功能

9. 登录、注册 - 基于 shiro

注册 登录 退出

10. 登录验证 - 基于 shiro

登录状态拦截器 安全验证

11. 事务管理

ForRESTController 对 createOrder 进行事务管理
等等 . . .

12. 缓存处理

全站数据通过 redis 进行了缓存

13. 动态静分离、反向代理

nginx

7. 产品展示

前台首页 前台产品页