从0开始做一个项目

动机: 处理日常生活的问题/有需求了/别人找到我要做一个... (需求)

分析: 我们要做一个什么样子的项目? (仿照, 找一些类似的网站)

整理: 可以先做一个简易版, 然后再网上增量开发

(1): 明确你的需求, 需要哪些功能? (电商类的网站, 注册功能, 登录功能, 首页部分, 分类页面商品预览, 商品详情页, 下单放入购物车)

PC端---模块:

注册

登录

首页

分类页

商品详情页

购物车页

(2): 如果暂时没有后端, 可以先写前端的静态页面. 如果有后端的话.

设计数据库 -- 表结构

首页模块涉及到的:

1. 省份表 (provinces)

2. 市区表 (cities)

3. 地区表 (areas)

4. 用户表 (member)

5. 购物车表 (shopcar)

6. 搜索表 (search) (保存用户搜索过的文字)

7. 一级分类表 (category\_first)

8. 二级分类表 (category\_second)

9. 三级分类表 (category\_thired)

10. 轮播图数据表 (banner)

11. 抢购表 (细分)

抢购活动 (flash\_sale)

每个活动下属的商品 (flash\_product)

12. 订单表 (看销量) -> 返回3-4个分类, 每个分类下返回3-4件商品

13. 订单表 (order\_v, order\_detail) (看评价分数) -> 返回8个商品 (评价排序)

order\_v (订单 - 商品) order\_detail (订单 -用户)

地址表address (因为下订单, 需要地址)

14. 首页模块表 (home) (下面铺设每一个大模块, 先记录都铺设哪些2级分类以及对应的下属的东西)

15. 猜你喜欢表 (guess\_and\_love)(记录你浏览过的商品, 后端代码取出你这个用户浏览过的商品, 再查询类似的给你返回)

分类商品预览页面

1. 商品表(goods\_list) (保存所有商品) (一对多的时候, 可以把多的那部分数据独立形成一个表)

商品评价表(goods\_eval)

商品收藏表(goods\_favorite)

商品图片表(goods\_image)

商品样式表(goods\_style)

2. 商品推荐橱窗表 (win\_location) (左下角)

一对多 (一件商品, 可以有多个用户去收藏), 分表(goods\_list商品表, goods\_favorite收藏表) 保存

所有图片地址(image)

商家表 (seller)

(3): 就可以根据划分的这些模块 -> 确定表名字 -> 用可视化工具 去navicat里手动创建这些表

(4): 设计每个表里的字段列名

根据页面上的内容, 来决定需要哪些列, 设计表里的字段, 以及对应的数据类型.

举例: goods\_list里: 来保存一件商品的数据

id: 一般作为主键+自增+int类型 序号而使用.

如果在页面里有一大段东西不想在分了, 可以把它当做一堆标签, 保存在数据库中. (标签字符串, 富文本字符串)

(5): 录入测试/正式数据

1. 手动录入数据 (效率极低)

2. 寻找数据 (省市区的数据, 可以百度搜sql导入省市区的数据, 会找到sql语句 / .sql文件 导入即可)

3. 研究任意一个语言的爬虫功能(nodejs, php, java), python爬虫. 爬虫下来的数据, 直接写入到数据库表里.

// 到此往上, 数据准备就绪

编写数据接口

接参, 调用数据库执行sql, 逻辑处理, 把最终结果返回给前端使用

1. 后台采用nodejs+express来编写api数据接口

2. 搭建后台的项目, 可以封装一些工具类来使用(或者从上一个项目复制下来工具类来使用哦) (Db.js / awaitTo.js/Result.js)

Db.js: 封装连接数据库, 执行sql语句, 返回执行后的结果

awaitTo.js 为了让await代码的异常, 可以在数组结构中返回

Result.js 是统一封装了对象的格式, 方便前端的判断

3. 我们就开始从首页开始编写数据接口

(1): 省份接口

4. 开发完接口, 为了测试接口是否可以直接使用 (可以不用写ajax, 也不建议使用浏览器地址栏来测试) 因为这2种测试, 以后没法形成文档保存下来

建议: 测试接口: Rest Client插件 + .http文件

5. 因为数据库表是下划线连接, 而前端人员通常使用驼峰标识, 所以在Success中处理一下返回的数据的key即可保证前后端都没问题.

(2): 市接口

(3): 区接口

(4): 热门搜索列表接口

是根据count的值排序, 选择前10个, 来返回给前端的, 这就是热门搜索

如果sql语句/后台逻辑代码中, 有动态的值, 需要前端配合传一个逻辑合理的参数给后台 (选择合适的请求方式), 如果sql语句中没有动态的值, 那就不需要传参.

(5): 录入搜索关键字

(6): 验证码接口:

采用svg-captcha来生成验证码图片, 并且在当前, 前端上设置了cookie(保存验证码正确的值)

无论是浏览器发起的请求 / Ajax发起的请求 / postman / restClient 都会被设置cookie

(7): 注册接口

uid代表用户的唯一标识, 数据库里保存的是图片的相对路径的地址 (图片文件保存在public静态资源目录下)

UUID的值, 作为uid (用户id)

UUID: 是封装的一个方法, 随机数+时间戳 生成了一个6位-4位-4位-4位-12位的数字字母的组合 (而且是唯一的不会重复)

注意: 后端/前端, 都可以生成uuid, 后端需要下载一个uuid的模块包

(7): 登录接口

注意: 用户在输入框里输入的是明文密码, 需要在前端的javascript中进行md5加密 (百度搜到js的md5加密.js的方法)

要返回一个jwt的token字符串, 用于替代用户+密码的使用, 在别的页面中, 可以在cookie上, 携带jwt给后台的任意接口, 后台如果需要uid, 这自己从cookie上提取即可.

jwt: JSON Web Token / Javascript Web Token (任何语言都可以使用jwt验证方式)

jwt组成:

header (标识加密使用的技术等)

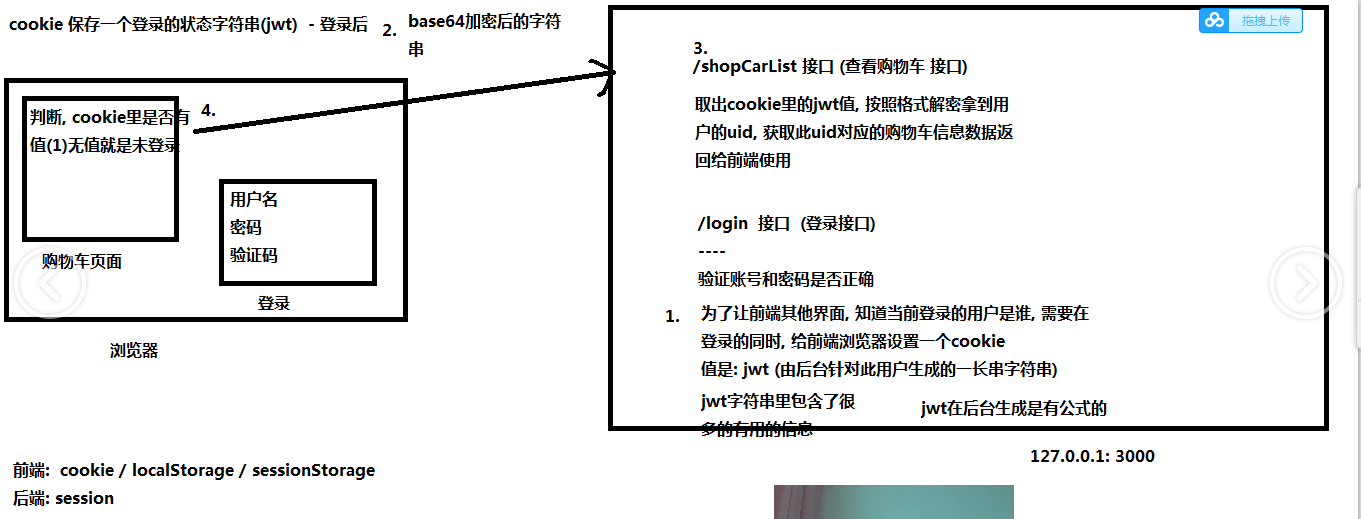
payload (数据载体, 就是jwt里装载的有用的数据, 比如uid, 生成的时间, 以及过期时间 (或者自己要放入的一些信息, 但是不要放敏感信息比如密码)

sign (识别码), 防止别人串改而使用的一个对比的条件值 (一般把前个连接起来并且加密生成它)

(8): 查询购物车接口

从cookie中提取jwt, 然后解密, 拆分, 拿到生成时间和有效时间 (毫秒单位), 和服务器当前时间对比, 判断有限期.是否已经过期

如果没过期, 这从jwt中提取uid, 查询对应的数据, 返回即可.



购物车数据, 发现都是ID, 没有具体的商品名字和商品图片等, 需要用各种ID去各个表里, 连接查询.

购物车数据换取:

1. 在页面上确定你都需要哪些数据 (图片, 样式名, 样式值, 商品名, 商品单价, 购物车里此商品的数量)

2. 你要对这些数据在哪些表里, 做到了如指掌 (多去找一找, 看一看), 确定字段.

3. 开始测试编写sql语句

(9) 搜索结果页面

SELECT

goods\_list.goods\_name,

goods\_list.goods\_price,

goods\_image.file\_name

FROM (goods\_list

LEFT JOIN goods\_image ON goods\_list.goods\_id = goods\_image.goods\_id)

WHERE goods\_list.goods\_name LIKE '%电脑%'

GROUP BY goods\_list.goods\_name

LIMIT 0, 16

1. 使用模糊查询的sql语句, 前端要把搜搜框的搜索文字传递给后台

2. 我们在后台做分页 (其实前台也可以做分页)

为什么要分页? 因为数据量太大, 而且前端一页也显示不下

前端传递页码, 和每页的数量(可选), 后端在limit上使用公式来限制返回的数据是哪一段!!!!

3. 前台做分页? 就是截取数组根据page的值 splice()

4. 细节操作, 只要前端页面上使用的属性字段, 所以在Select后面不要写\*, 只写需要的字段名即可

多表联查时, 在5.7.28以后, 会在mysql配置中, 默认打开groupby的限制(每个select后面的字段, 都得出现在gourp by的里面)

如果不出现查询时就会爆出如下的错误:

ERROR 1055 (42000): Expression #1 of SELECT list is not in GROUP BY clause and contains nonaggregated column 'xxx.xxx.room\_id' which is not functionally dependent on columns in GROUP BY clause; this is incompatible with sql\_mode=only\_full\_group\_by

如何关闭这个限制?

在数据库安装的目录的my.ini文件 (数据库配置文件)

在末尾追加如下这行

sql\_mode=STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION

需要在管理工具中->服务 -> mysql -> 重启服务 (让新的配置生效)

以销量排序:

SELECT

goods\_list.goods\_id,

goods\_list.goods\_name,

goods\_list.goods\_price,

count(order\_v.goods\_id) as sellCount

FROM goods\_list

LEFT JOIN order\_v ON goods\_list.goods\_id = order\_v.goods\_id

WHERE goods\_list.goods\_name LIKE '%休闲%'

GROUP BY goods\_list.goods\_id

ORDER BY sellCount DESC

解释: 2表联查: 商品表和订单表, 每件商品会出现在不同的订单中, 所以count()统计的是此商品id, 在订单中出现的次数count, 起别名sellCount

Group By 是要统计每件商品的销量, 所以条件是商品的goods\_id

Order By 以销量排序.

最后: 前端如何做的分页? 才能达到比较优秀的效果? (用变量在后台默默的缓存别的页面的数据

前端加入变量缓存,网页刚打开时, 使用ajax请求第一页和第二页的数据分别请求回来.

因为用户点击下一页, 我们从变量里取值, 而非ajax等待请求的数据, 所以速度会非常快, 给用户的体验也非常好. 每次多请求一页的数据.

(10): 分类结果接口

搜索结果页面不需要铺设分类部分的标签, 分类页面其实就多了一个参数叫thiredId传给后台, 后台可以复制粘贴搜索结果的接口里的代码改装一下即可.

(11): 首页-轮播图接口

直接返回banner表里的数据给前端

加料: 数据库里记录的图片地址, 都是相对路径, 前端没法直接使用(如果能直接使用, 是因为前端项目和后端项目在一个express工程中, ip和端口是一样的而且image\_source就在public下面, 所以你可以直接访问)

但是: 如果前后端分离, 不在一个express工程下, 那么图片的地址就得给前端一个完整的 ip+端口+路径, 这样前端img的src才能显示

以后, 后台返回给前端的路径, 尽量都是绝对路径, 这样就不害怕找不到资源了

如果遇到一个变量频繁使用, 而且以后还可能被修改, 就把它抽离出来, 封装起来多次调用 (前后端都是这个意思)

CONCAT聚合函数的作用:

// CONCAT的作用: 是在查询结果时, 给某个字段进行拼接值

// CONCAT(在前面拼接的字符串, 查询的字段) as 给值起个别名

const [err, arr] = await Db.query(`SELECT CONCAT('${staticPath}', coverimg) as coverimg FROM banner`);

(12)首页-限时活动

涉及的表 flash\_sale, flash\_product, goods\_list, order\_v, goods\_image (接口的数据来自于这5张表)

1. 先查询前2个限时活动的ID

2. 循环这2个限时活动, 把每个限时活动id, 换出下属的多个商品id对象 goodsArr

3. 循环商品数组, 把每个商品的对象, 利用商品\_id去查询商品的名字...商品销量, ...商品的图片, goodsInfoObj, file\_name

4. 再把信息串联起来, 因为都是数组套对象的格式, map循环时, 对象和数组里的对象都是引用关系, 你给对象添加属性, 就是在给数组里的对象添加属性, 所以串联起来以后, 最外边只要把arr返回给前端, 那就把这一串串的数据返回给前端了.

(13) 首页 -- 人气好货

1. 查询的是eval\_start 大于等于3的都算好评, 聚合查询, 对每件商品来查询对应的好评数量count(\*)来统计, 好评数在前的就是人气好货 ,所以还要排序, 但是首页上只需要8个, 所以再加limit的条件

SELECT

goods\_id,

count(\*) as goodEvalCount

FROM goods\_eval

WHERE eval\_start >= 3

GROUP BY goods\_id

ORDER BY goodEvalCount DESC

LIMIT 0, 8

2. 循环每个商品对象, 提取商品ID, 换区商品的名字和图片, 串到arr数组的对象上返回给前端

(14) 首页 - 销量排行榜

1. 查询order\_v表, 统计每件商品的销量, 用销量排序, 取出4个三级分类的id

2. 对每个三级分类的ID, 去换取3件商品 串起来返回给arr即可

(15) 首页 - 各个模块

1. 记录了一个每块二级分类ID, 在home表中

2. 二级分类ID 去换取下属的三级分类数据, 以及二级分类的4件商品即可

let a = await Promise.all() a的值是Promise.all里面所有的小Promise的resolve的结果组成的数组 (规定格式)

let a = Promise.all() a的值就是一个大的Promise对象

a.then()来接收结果

(16) 首页 - 猜你喜欢

未登录的话, 随机返回一些数据

已经登录的话, 从guess\_and\_love表, 提取它喜爱的三级分类下的一些商品数据 (如果没有数据, 还是随机返回一些数据)

ORDER BY RAND() 作用: 给每条记录分配一个随机数, 然后用随机数来影响当前这一条的数据的排列顺序.

如果有猜你喜欢里的数据, 随机返回一个三级分类ID, 然后根据这个三级分类ID, 随机返回20件商品给前端铺设, 猜你喜欢...

(17) 商品 - 详情页

判断现在cookie里有没有jwt, 提取uid, 然后录入到guess\_and\_love(此用户的浏览历史)

1. 是一套标签, 套万套不同的数据

2. 此页面怎么知道显示哪件商品的详情数据呢? 在前端点击跳转到商品详情页的时候, 把商品的id(goods\_id), 传递到商品详情页的html去

知识点: 前端跨网页html传参: (?后面拼接, cookie, sessionStorage, localStorage)

3. 详情页拿到goods\_id 以后, 就要调用商品详情的接口, 来换取数据

(18) 橱窗

直接查询win\_location表里的6件商品的信息, 返回给前端使用即可.