|  |
| --- |
| **时见科技电商系统**  **部署文档** |

**时见科技有限责任公司**

**文档日期：二零一九年四月**

**修改记录**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **编号** | **日期** | **描述** | **版本** | **作者** | **审核** |
| 001 | 2019/04/10 | 创建 | V1.0 | 陈礼忱 |  |
| 002 | 2019/05/05 | 增加用户模块 | V1.1 | 陈礼忱 |  |
| 003 | 2019/06/05 | 增加分类管理 | V1.3 | 陈礼忱 |  |
| 004 | 2019/08/05 | 增加商品模块 | V1.4 | 陈礼忱 |  |
| 005 | 2019/08/06 | 增加购物车模块 | V1.5 | 陈礼忱 |  |

本文档中所包含的信息属于机密信息，如无时见科技有限公司的书面许可，任何人都无权复制或利用。

®Copy Right 2019 by TimeSee

目录

[1数据库 4](#_Toc17632)

[1.1数据库初始化 4](#_Toc5908)

[2 项目初始化 8](#_Toc11406)

[2.1 IDEA项目初始化 8](#_Toc8844)

[2.2 IDEA项目运行 12](#_Toc19777)

[2.3 git初始化 12](#_Toc22344)

[2.4 maven的文件配置 17](#_Toc18995)

[2.5 项目包结构初始化 17](#_Toc28703)

[2.6 mybatis-generator配置 18](#_Toc11586)

[2.7 mybatis-plugin安装 21](#_Toc10326)

[2.8 mybatis-pagehelper分页插件安装 22](#_Toc8721)

[2.9 提交代码到git 22](#_Toc11435)

[2.10 Restlet Client和FE助手 24](#_Toc15208)

[2.11 Nignx反向代理服务器 24](#_Toc452)

[3 用户模块 37](#_Toc7272)

[3.1 门户接口 37](#_Toc29442)

[3.1.1 登录 37](#_Toc27791)

[3.1.2 用户名验证 38](#_Toc29964)

[3.1.3 注册 38](#_Toc29851)

[3.1.4 忘记密码问题获取 39](#_Toc395)

[3.1.5 提交问题答案 40](#_Toc29649)

[3.1.6 忘记密码的重置密码 40](#_Toc24047)

[3.1.7 登录状态重置密码 41](#_Toc30772)

[3.1.8 获取用户信息 42](#_Toc3383)

[3.1.9 登录状态更新个人信息 43](#_Toc6569)

[3.1.10 获取当前登录用户的详细信息，并强制登录 44](#_Toc27917)

[3.1.11 退出登录 45](#_Toc14119)

[3.2 后台接口 46](#_Toc16567)

[3.2.1 后台管理员登录 46](#_Toc3672)

[3.3 学习目标 46](#_Toc16563)

[3.3.1 横向越权纵向越权的安全漏洞 46](#_Toc2395)

[3.3.2 MD5明文加密及增加salt值 47](#_Toc15341)

[3.3.3 Guava缓存的使用 47](#_Toc15240)

[3.3.4 高复用服务响应对象的设计思想及抽象封装 47](#_Toc32756)

[3.3.5 Mybatis-plugin的使用技巧 47](#_Toc3430)

[3.3.6 Session的使用 47](#_Toc10144)

[3.3.7 方法局部演进 47](#_Toc22832)

[4 分类管理模块 47](#_Toc11259)

[4.1 分类管理模块开发概述 47](#_Toc1177)

[4.2 学习目标 47](#_Toc12659)

[4.3 数据库设计 48](#_Toc18871)

[4.4 分类管理模块接口 48](#_Toc7071)

[4.4.1 获取品类子节点（平级） 48](#_Toc12348)

[4.4.2 增加节点 50](#_Toc1048)

[4.4.3 修改品类名字 50](#_Toc348)

[4.4.4 获取当前分类ID以及递归子节点categoryId 51](#_Toc21816)

[5 商品模块 52](#_Toc5146)

[5.1 功能介绍 52](#_Toc4033)

[5.1.1 前台功能 52](#_Toc31957)

[5.1.2 后台功能 53](#_Toc15025)

[5.2 学习目标 53](#_Toc16961)

[5.2.1 FTP服务的对接 53](#_Toc14039)

[5.2.2 springMVC文件上传 53](#_Toc2870)

[5.2.3流读取Properties配置文件 53](#_Toc15327)

[5.2.4抽象POJO、BO、VO对象之间的转换关系及解决思路 53](#_Toc5123)

[5.2.5 joda-time快速入门 54](#_Toc13039)

[5.2.6静态块 54](#_Toc28437)

[5.2.7 Mybatis-PageHelper高效准确地分页及动态排序 54](#_Toc785)

[5.2.7 Mybatis对List遍历的实现方法 55](#_Toc29760)

[5.2.7 Mybatis对where语句动态拼装的几个版本演变 55](#_Toc17914)

[5.3 数据表设计 55](#_Toc6892)

[5.4 接口设计 55](#_Toc20457)

[5.4.1 前台商品接口设计 55](#_Toc7912)

[5.4.2 后台商品接口设计 58](#_Toc3003)

[6 购物车模块 66](#_Toc3262)

[6.1 功能介绍 66](#_Toc29802)

[6.1.1 前台功能 66](#_Toc31163)

[6.2 学习目标 66](#_Toc32639)

[6.2.1购物车模块设计思想 66](#_Toc3326)

[6.2.2如何封装一个高复用的购物车核心方法 66](#_Toc10321)

[6.2.3解决浮点型商业运算中丢失精度问题 66](#_Toc27219)

[6.2.4解决浮点型商业运算中丢失精度问题 67](#_Toc2568)

[6.3 数据表设计 67](#_Toc21676)

[6.4 接口设计 68](#_Toc9897)

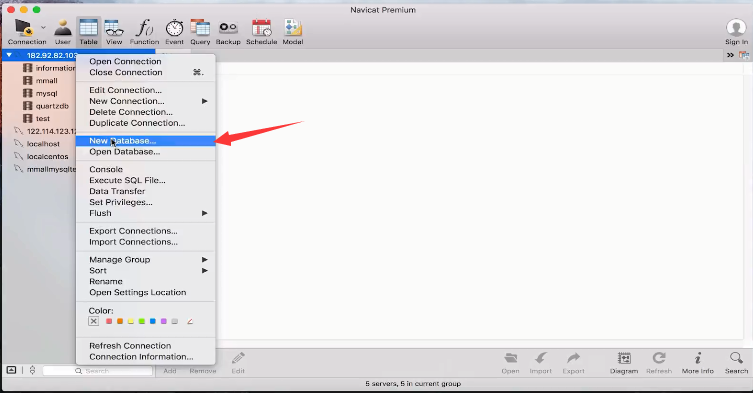
[6.4.1 前台购物车接口设计 68](#_Toc19813)

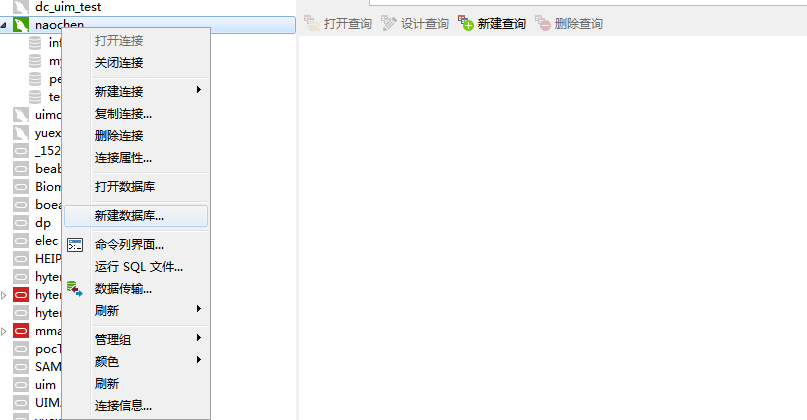
[7 常见错误 77](#_Toc11210)

# 1数据库

## 1.1数据库初始化

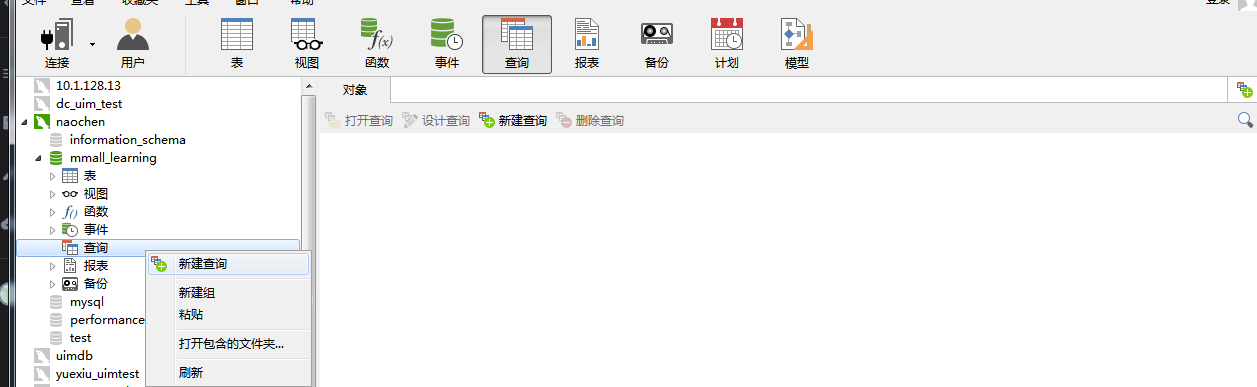
1、右键->New DataBase



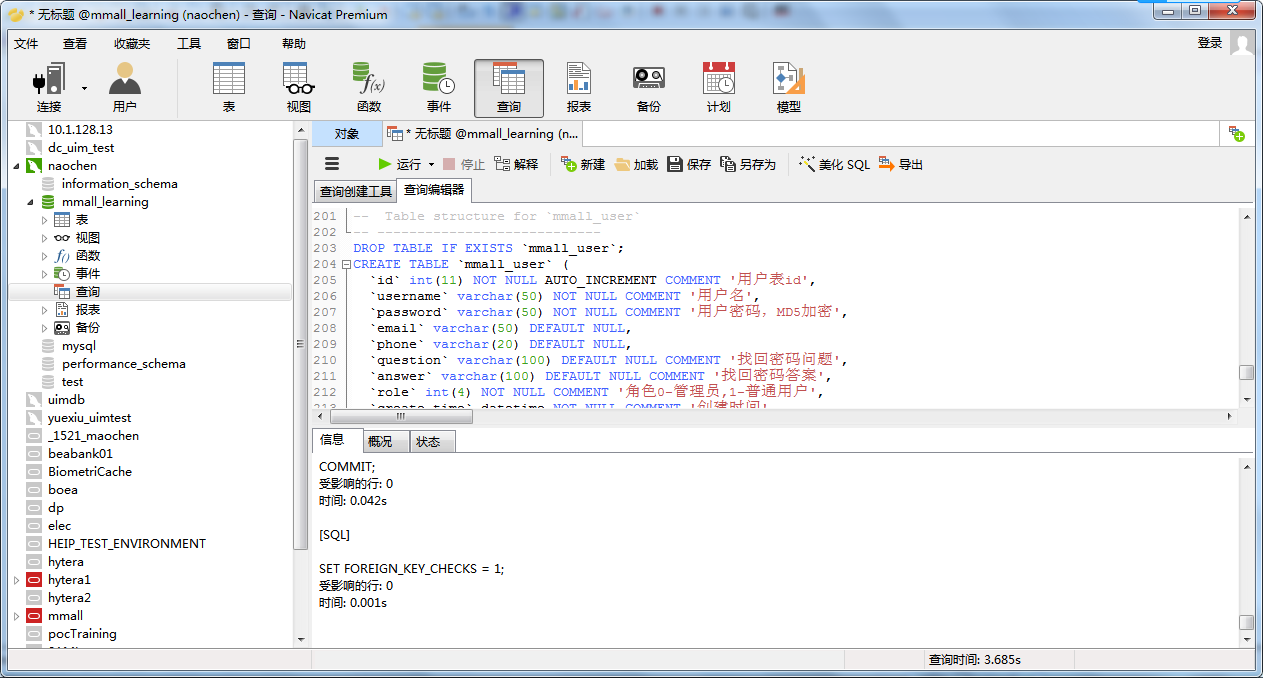




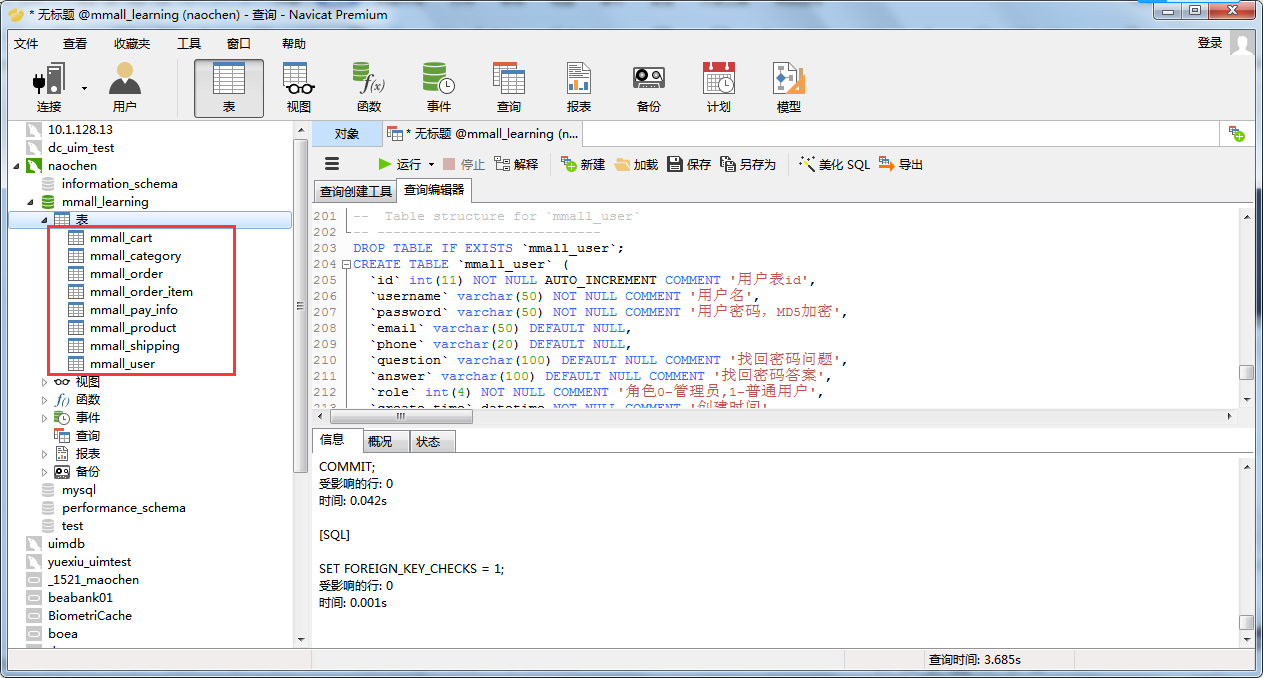
新建一个查询：



将所有的sql语句复制粘贴到查询中，然后执行



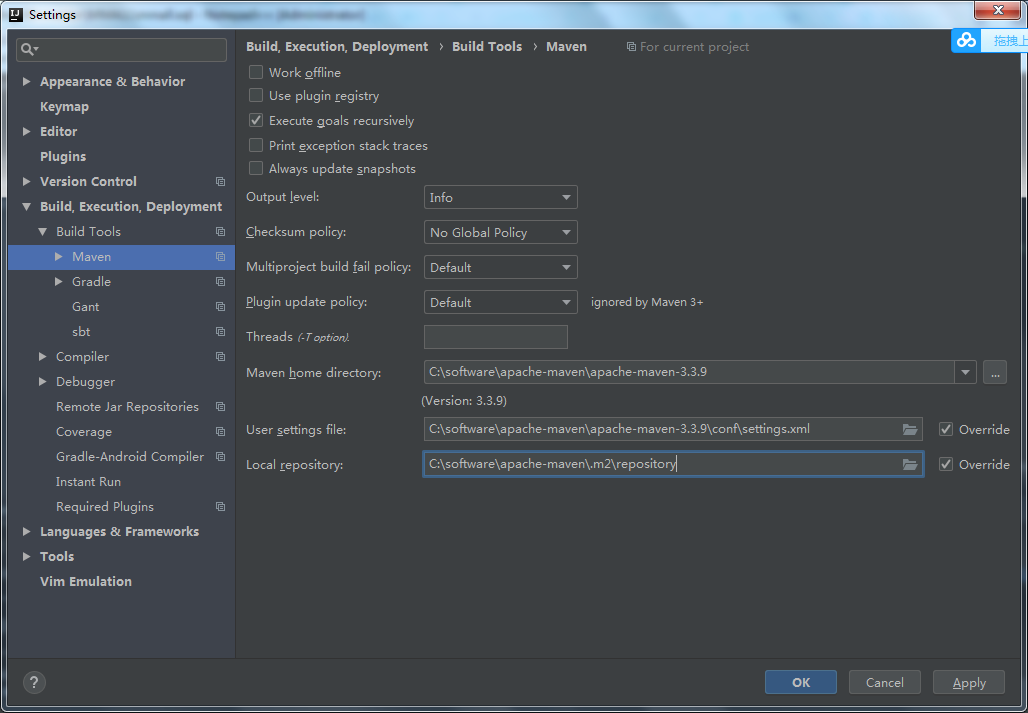
可以看到表已经创建成功



# 2 项目初始化

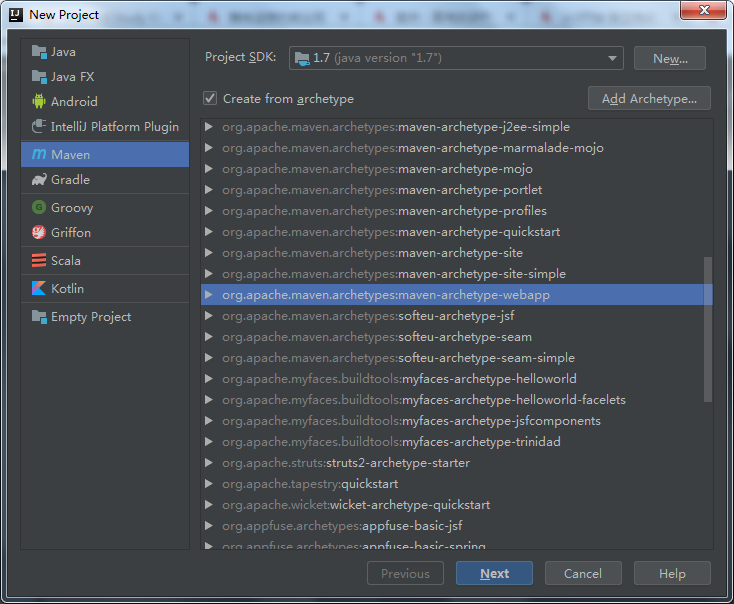
## 2.1 IDEA项目初始化

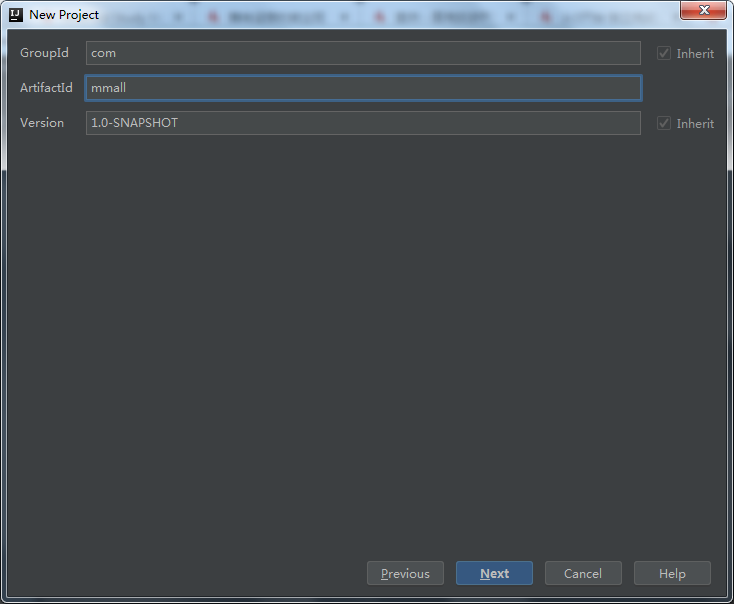
Maven配置：

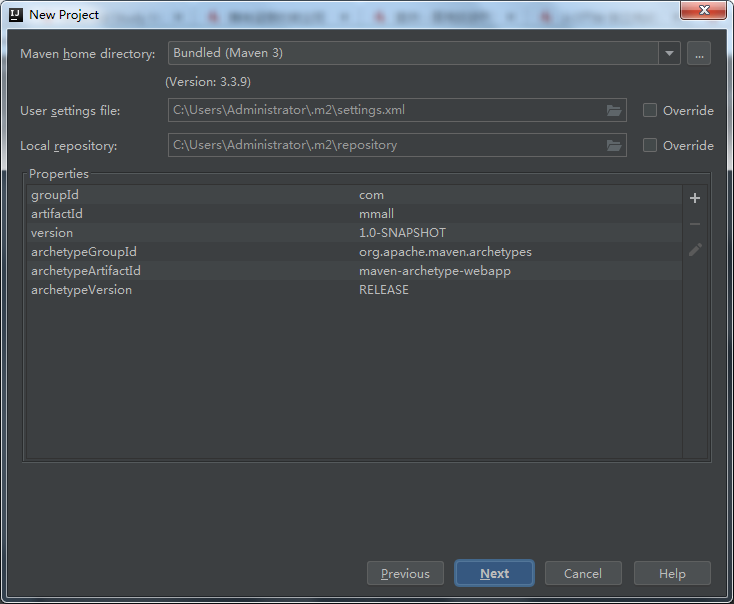


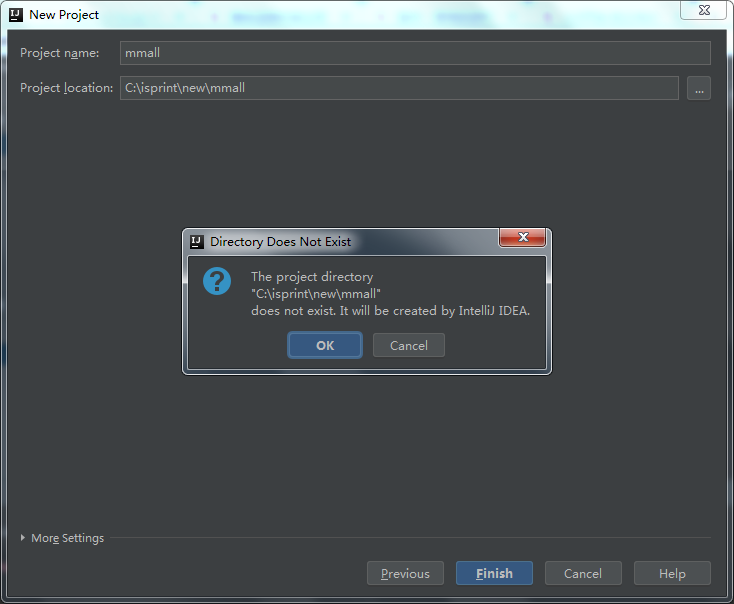


选择maven的archetype-webapp





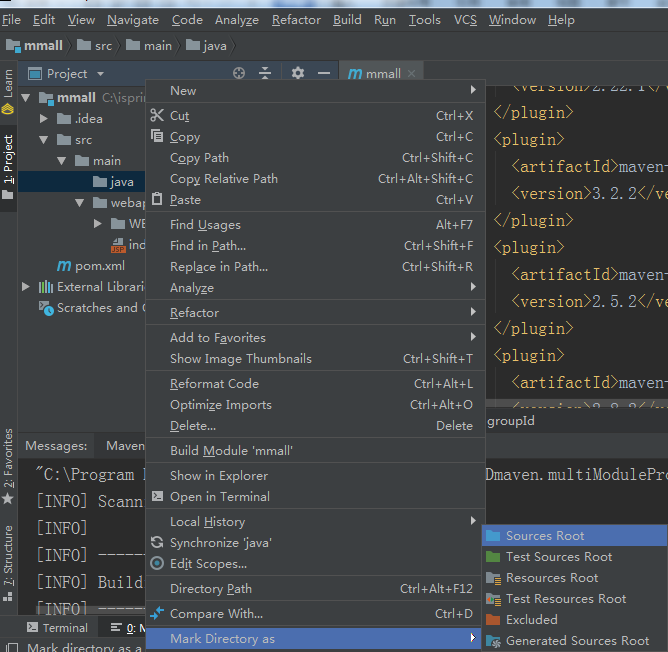




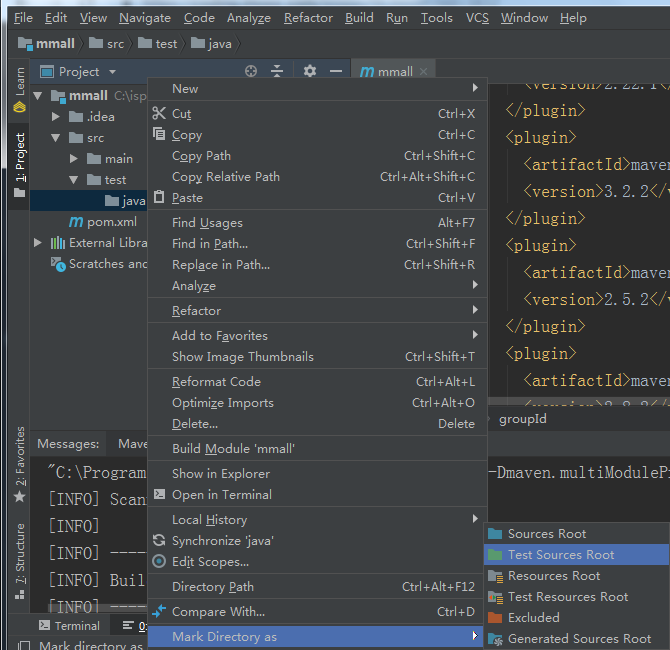
项目新建成功：



在src/main目录下创建一个Directory，然后设置为Source Root



然后再src目录下创建一个test文件夹，然后再test文件夹下创建一个java文件夹，设置为Test Sources Root

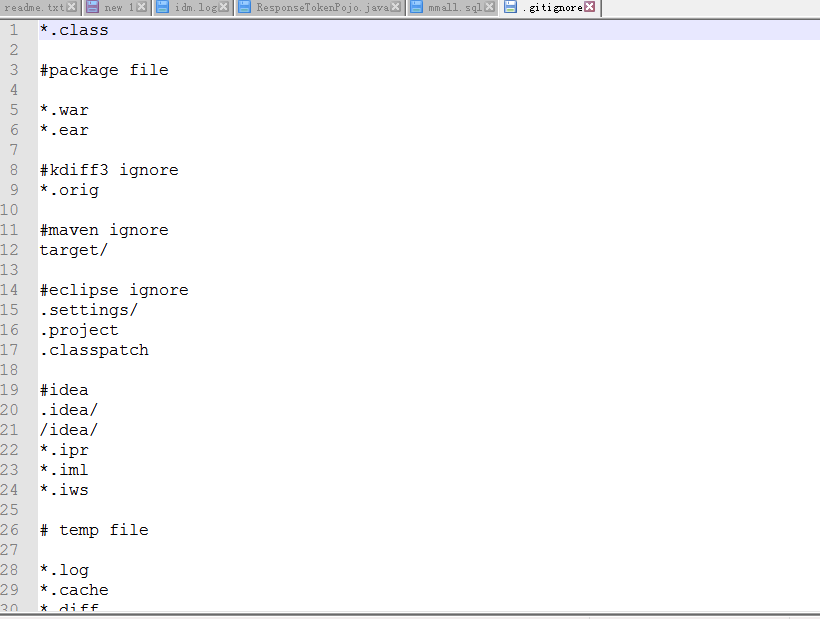


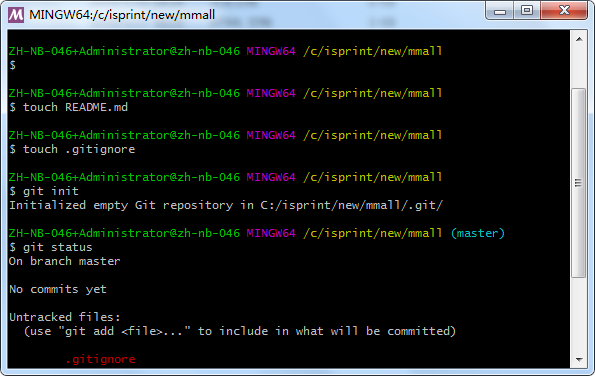
## 2.2 IDEA项目运行

## 2.3 git初始化

创建一个README.md文件，创建一个.gitignore文件

并且在.gitignore文件中添加如下内容



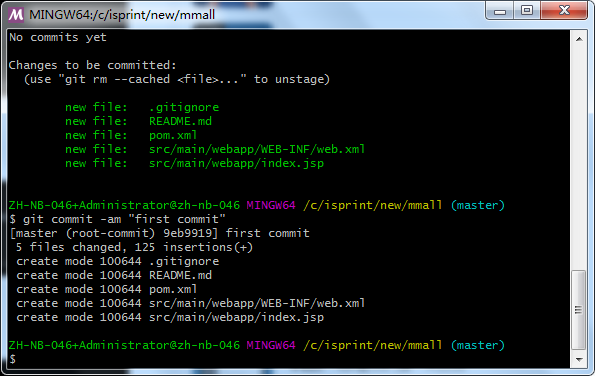


使用 git init对工程进行初始化

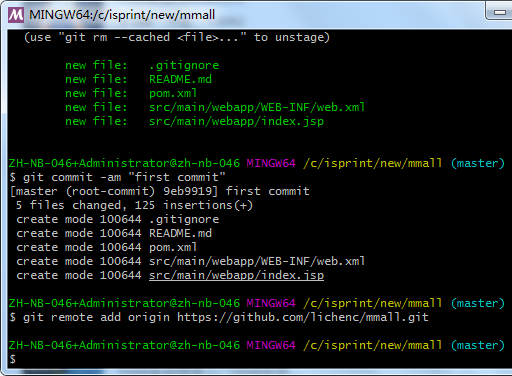
使用git statuc对工程进行查看状态

使用git add . 将所有的文件添加

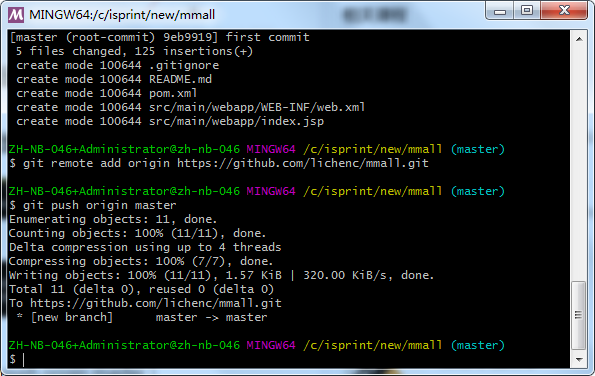
使用git commit -am “first commit” 提交到本地仓库



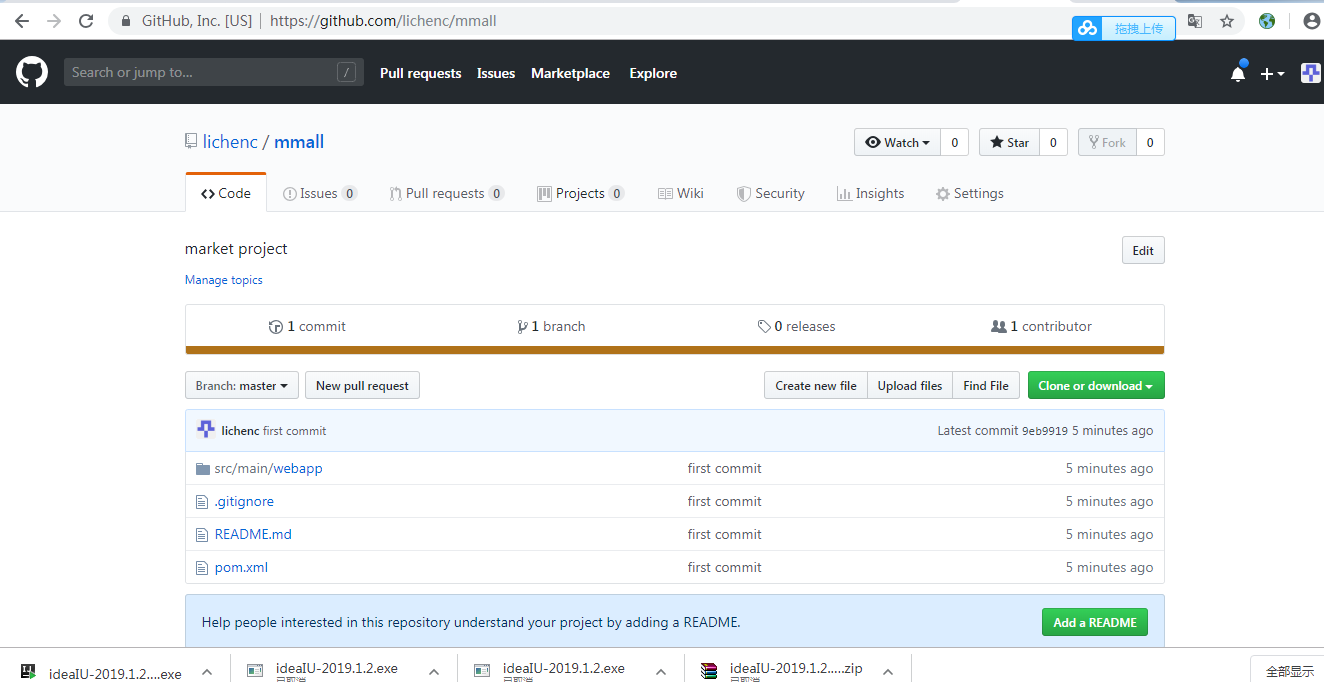
增加远程仓库别名



提交到远程仓库

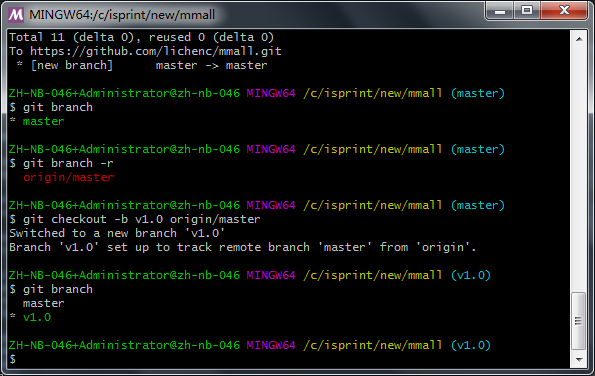


然后刷新github上的页面

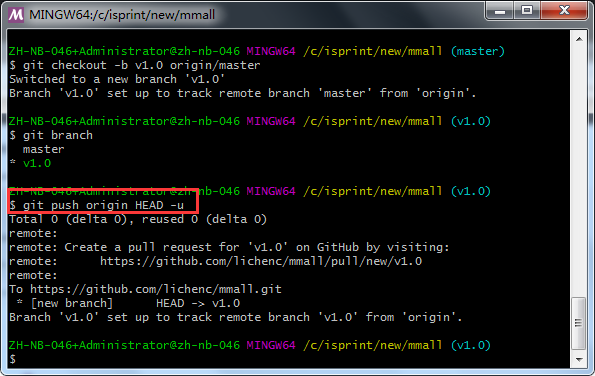


我们采用分支开发，主干发布的方式：

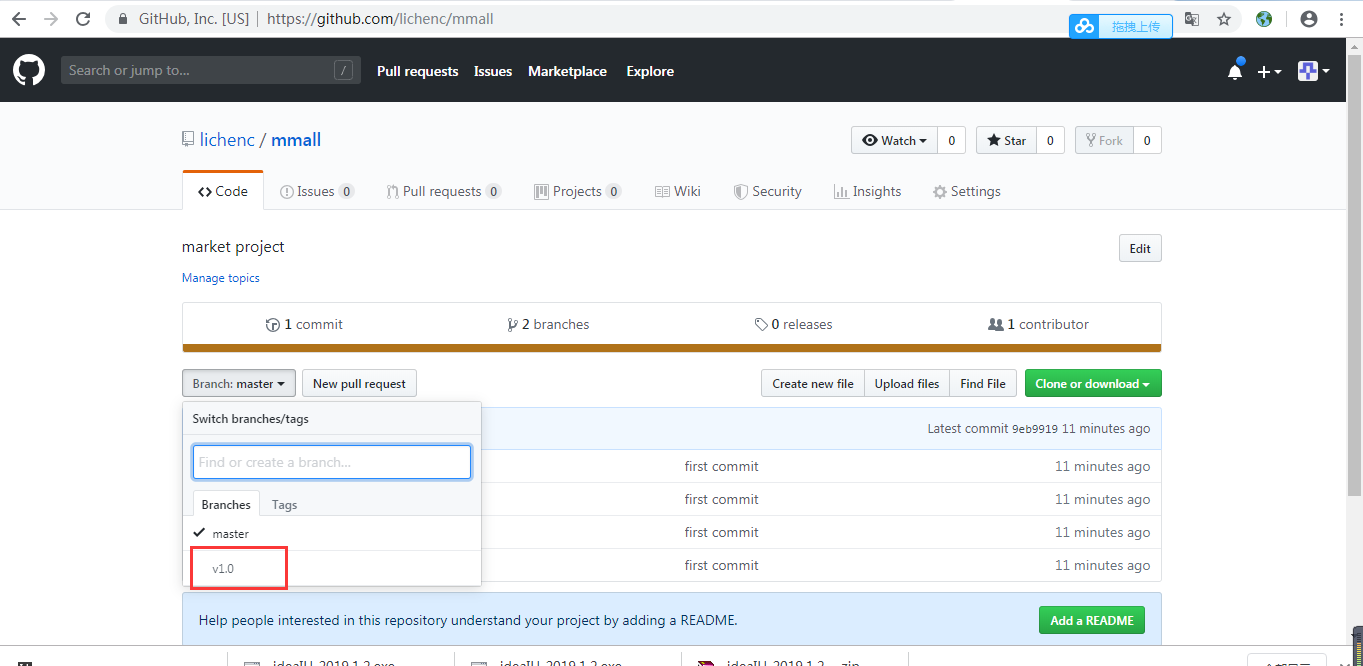
创建一个分支 V1.0,



将分支推送到远程仓库上



查看远程仓库的状态



## 2.4 maven的文件配置

Maven提供的中央仓库，http://search.maven.org/

## 2.5 项目包结构初始化

在main/java下创建以下包名

controller 是控制层

service是业务层

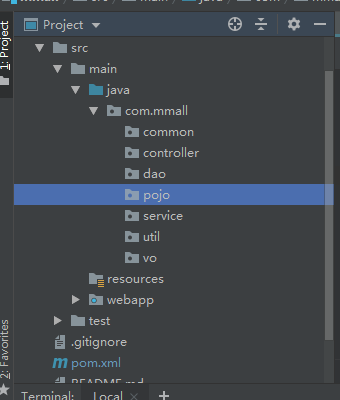
dao是与数据库交互的持久层

vo是view object

pojo是数据库表的对象

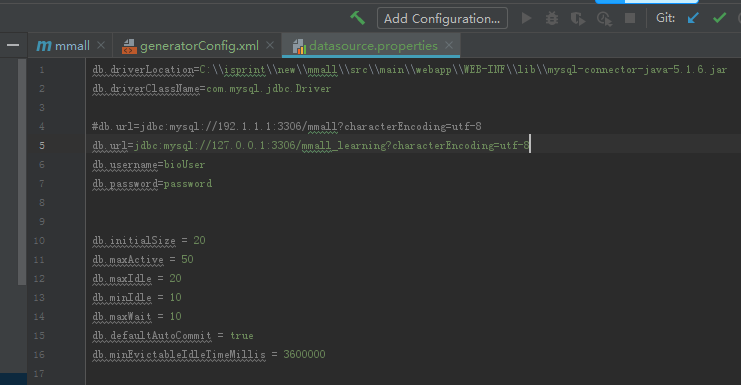
common是常用类或者常量

util是工具类存放的包

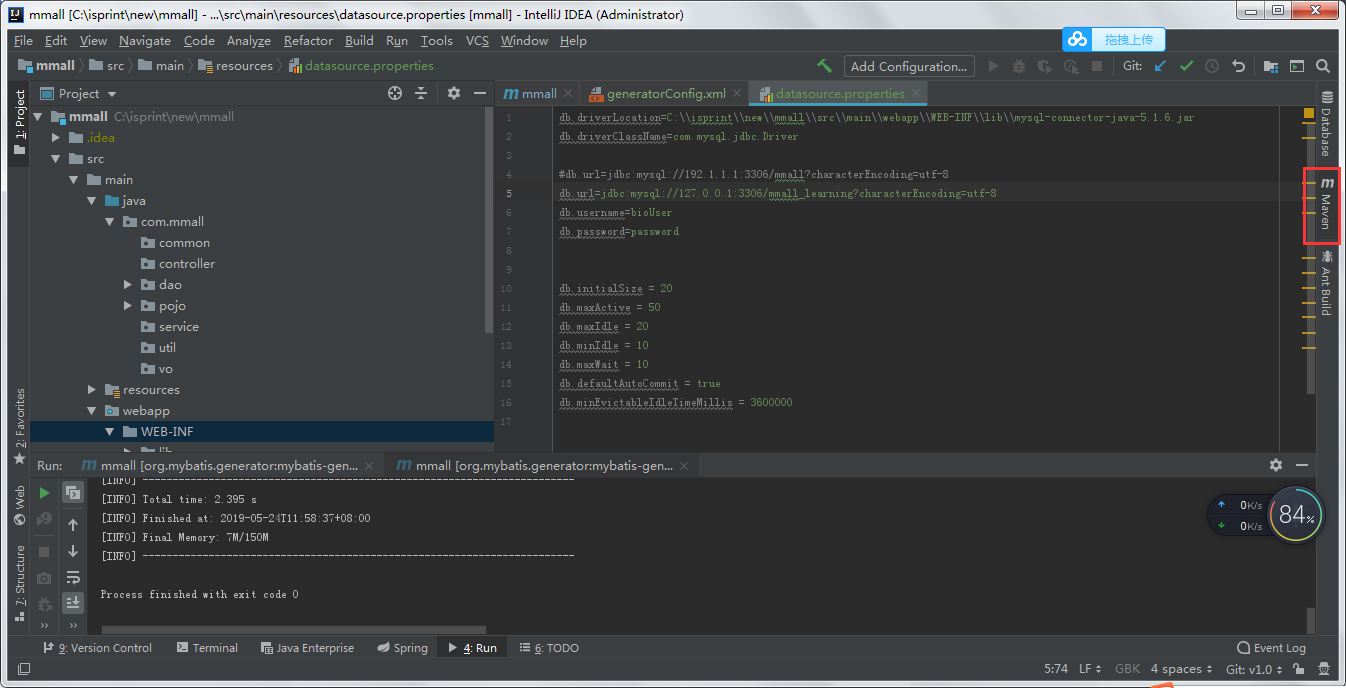


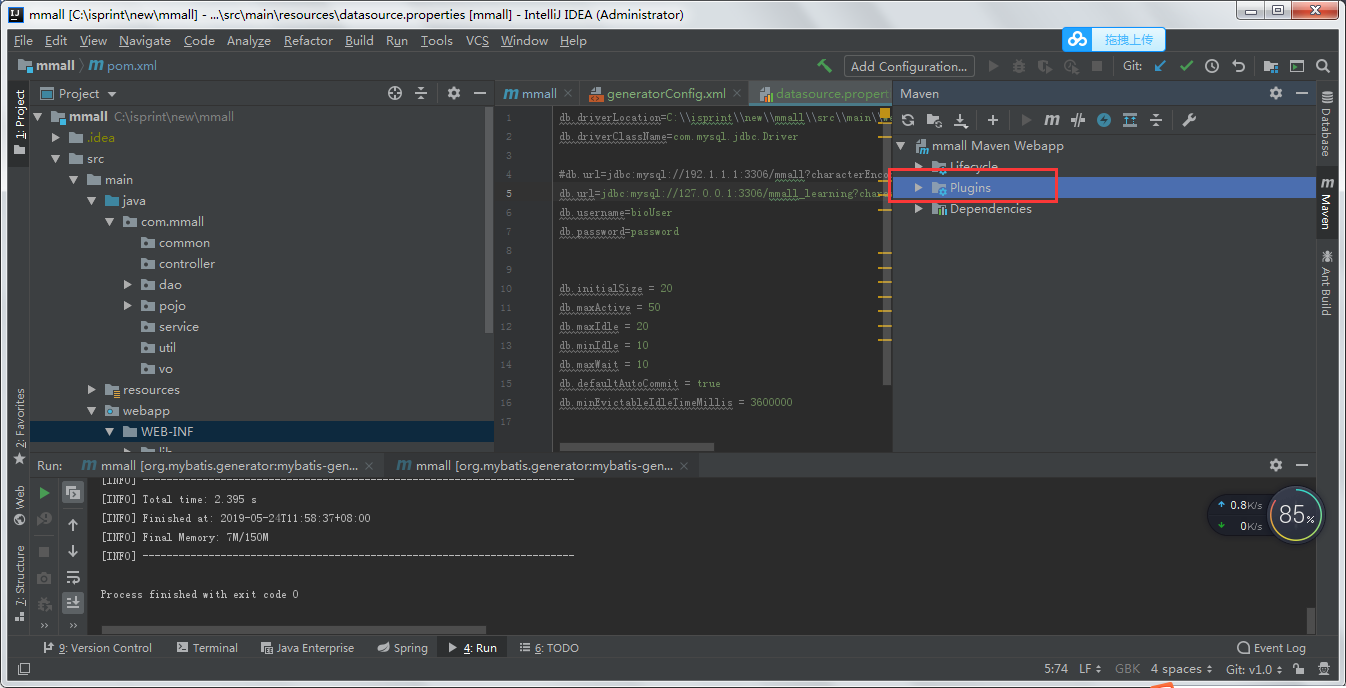
## 2.6 mybatis-generator配置

将generatorConfig.xml文件和datasource.properties放到resources目录下，然后配置datasources.properties文件

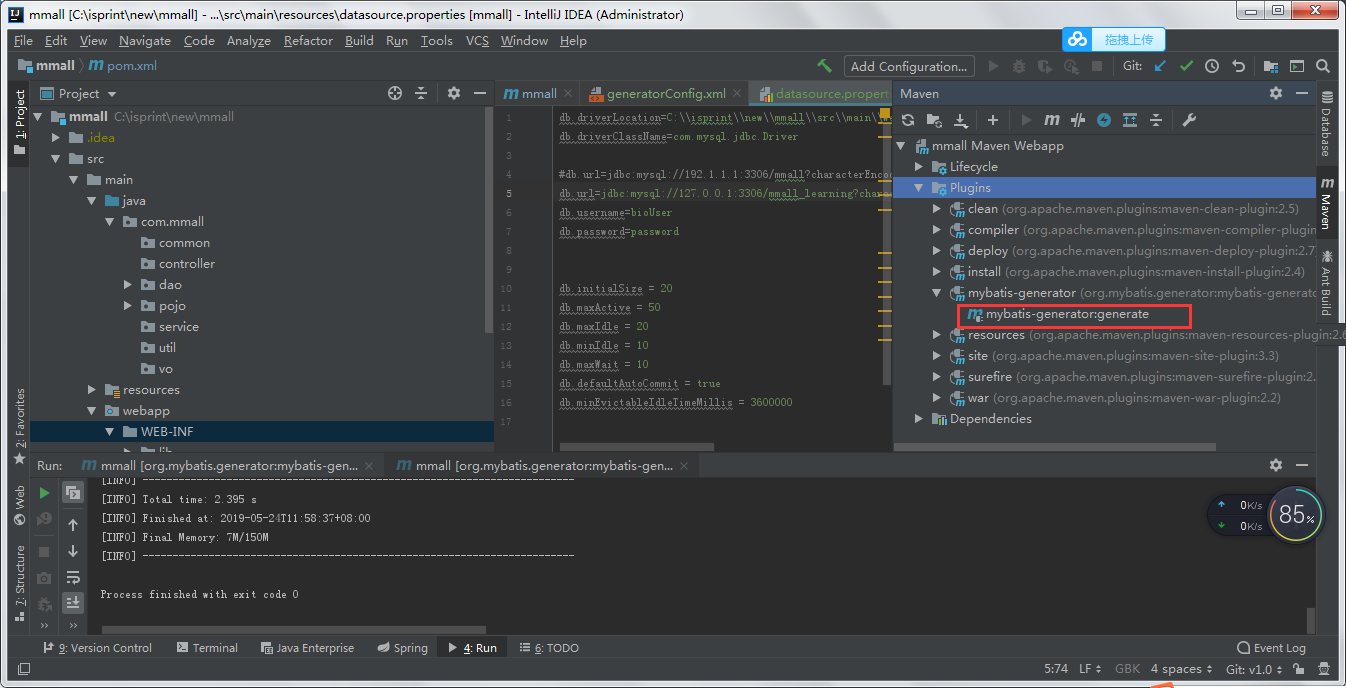


然后在idea的右边的maven工具中点击plugins

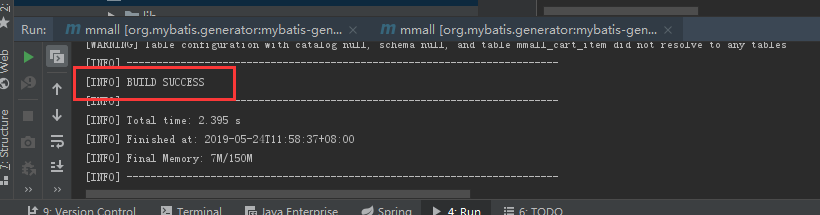




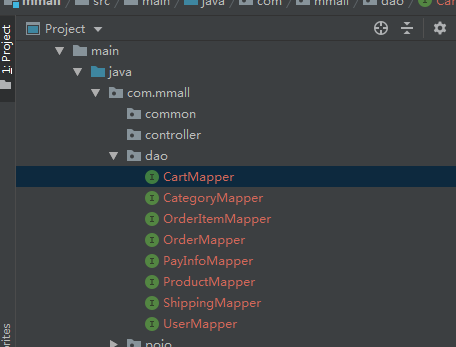
然后双击mybatis-generator : generate

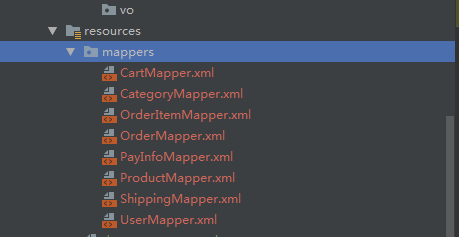


双击后编译成功

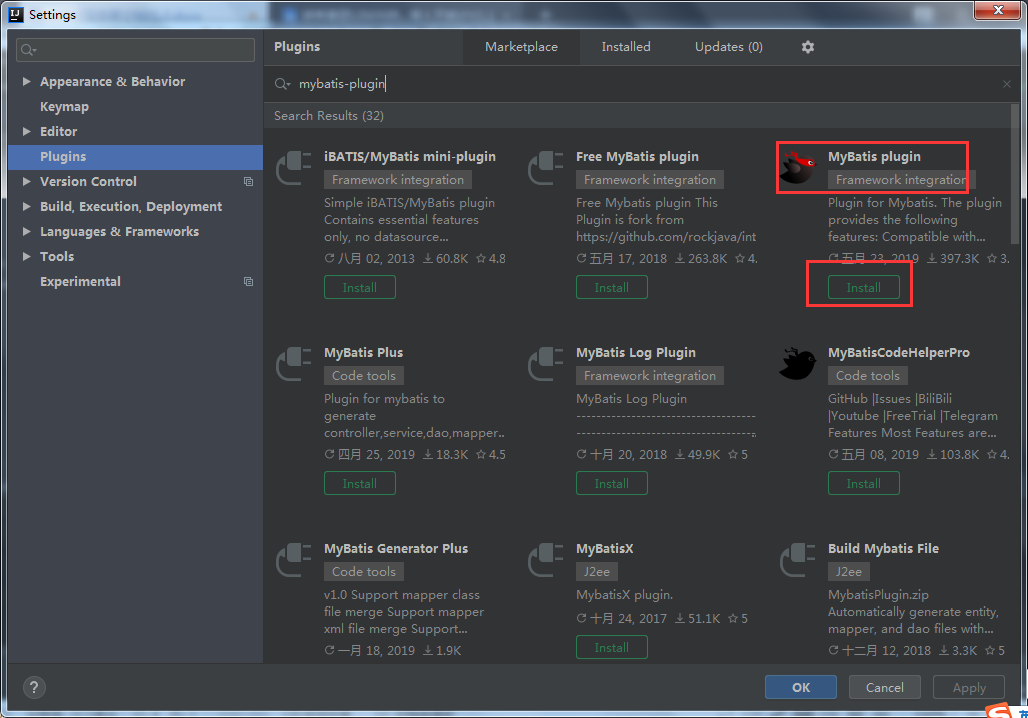


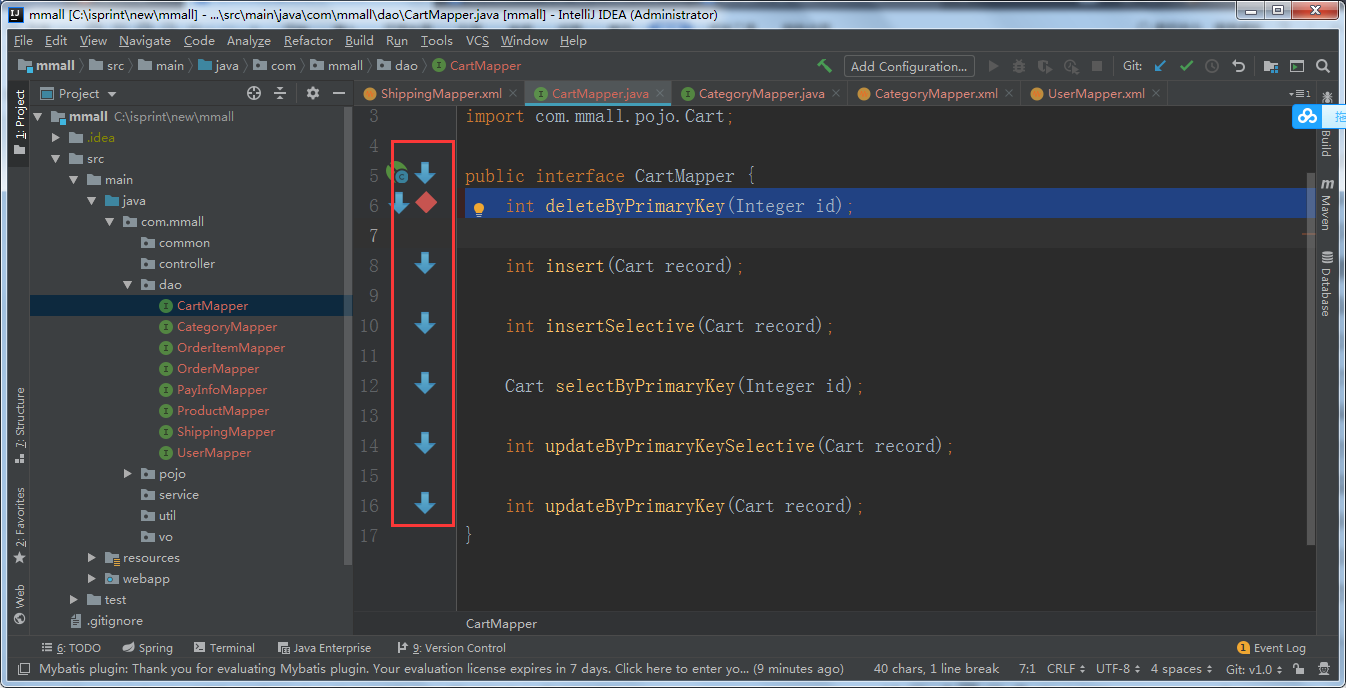
在java目录下的dao和pojo将会自动生成与数据库表相对应的文件，还有resources下也有生成mappers目录

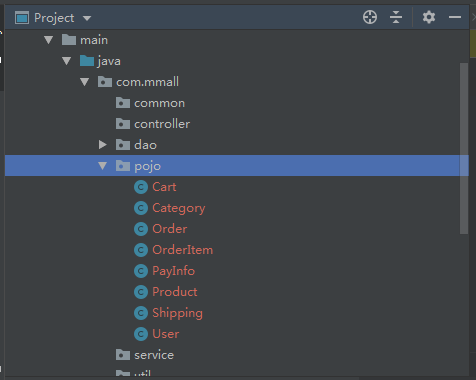




## 2.7 mybatis-plugin安装

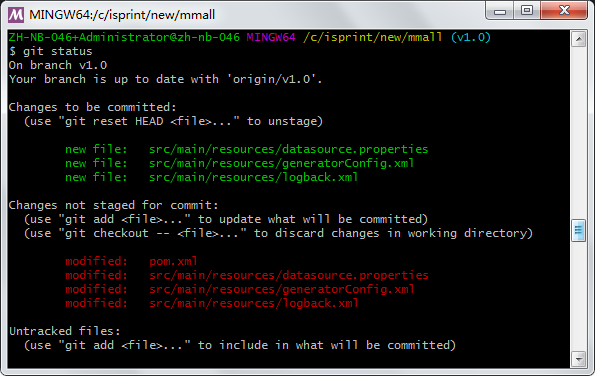


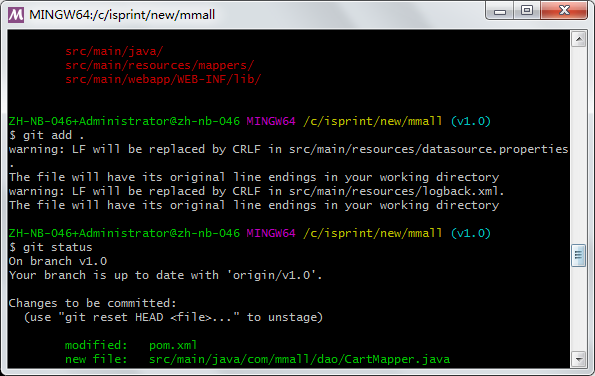




## 2.8 mybatis-pagehelper分页插件安装

## 2.9 提交代码到git







提交到github上

## 2.10 Restlet Client和FE助手

到google的商店中下载Restlet Client 和PE助手

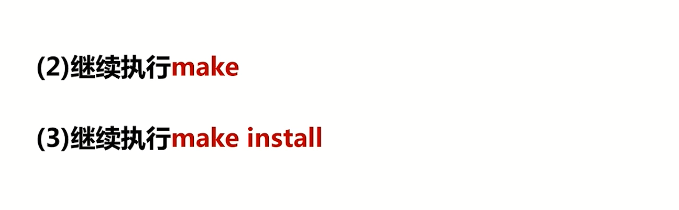
## 2.11 Nignx反向代理服务器



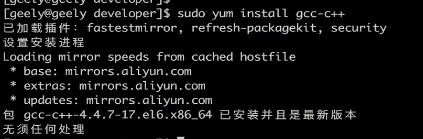




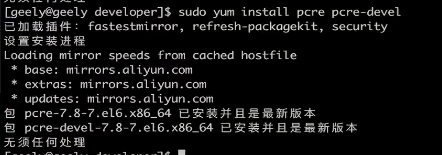




sudo yum install gcc-c++



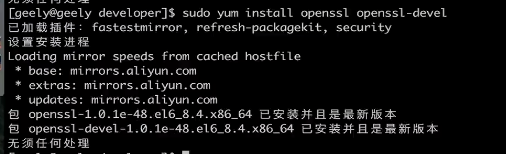
sudo yum install pcre pcre-devel



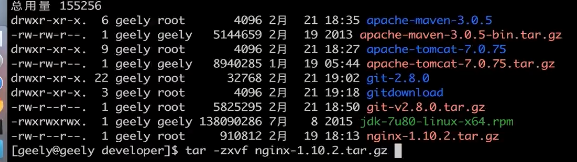
sudo yum install zlib zlib-devel



sudo yum install openssl openssl-devel



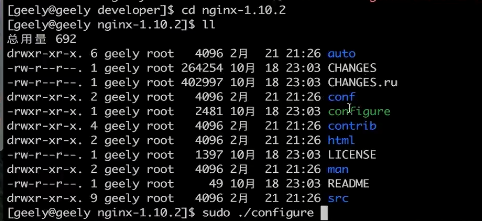
Tar -zxvf ngnix-1.10.2.tar.gz



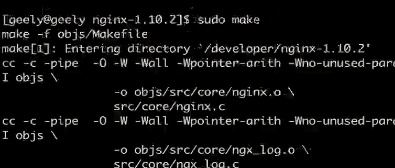
cd nginx-1.10.2

Ll

sudo ./configure



Sudo make

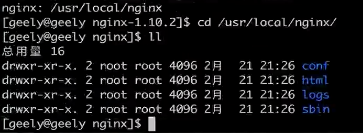


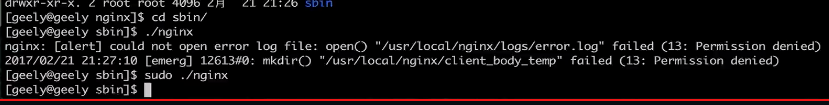
安装

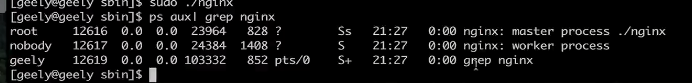
Sudo make install



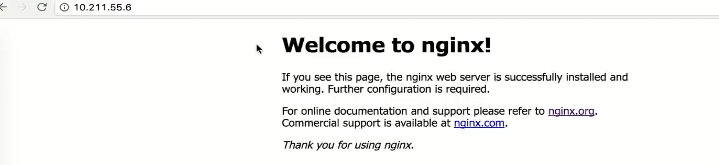








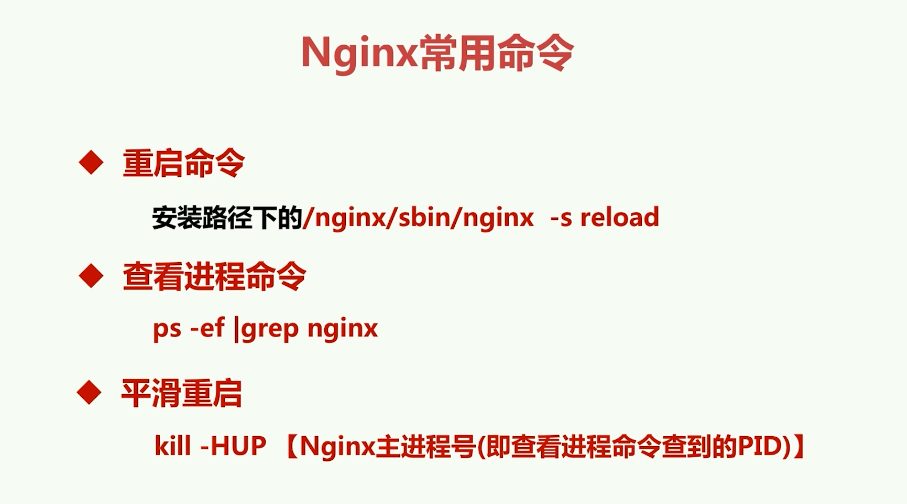
用http://localhost/访问















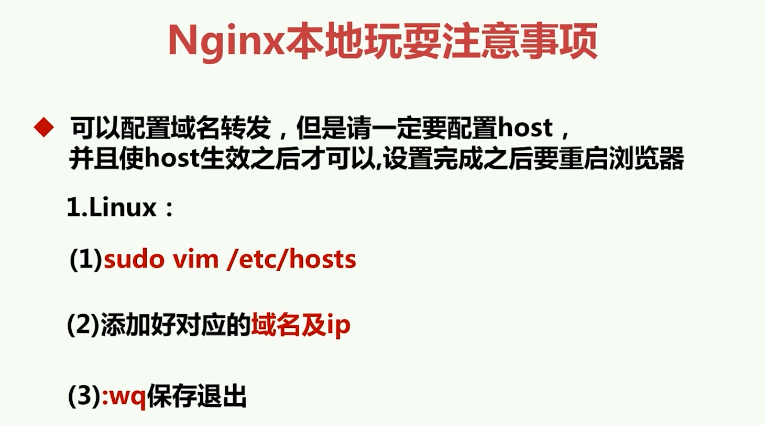




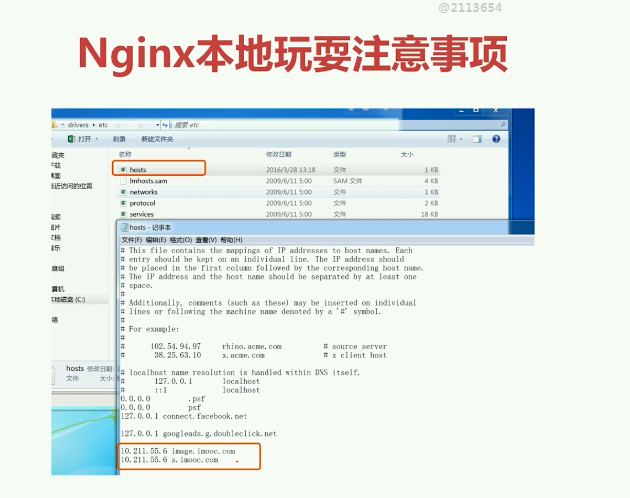


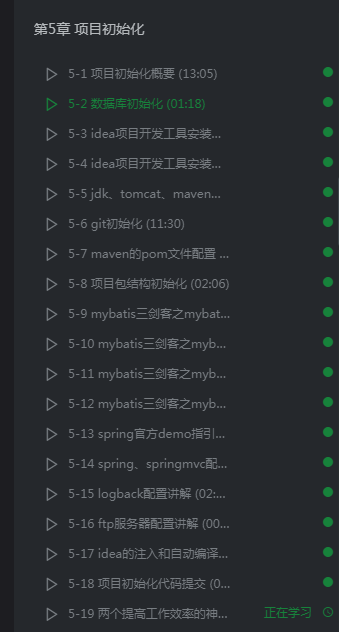












# 3 用户模块

## 3.1 门户接口

### 3.1.1 登录

接口设计：

**/user/login.do** post代码需要post方式请求，开发get，方便测试

**request**

Username ,password

**Response**

**Fail**

{

“status” : 1 ,

“msg” : ”密码错误”

}

**Success**

{

“status” : 0,

“data” : {

“id” : 12,

“username” : ”aaa”,

“email” : [”aaa@163.com”,](mailto:\”aaa@163.com\”,)

“phone”:null,

“role” : 0,

“createTime” : 1479048325000,

“updateTime” : 1479048325000

}

}

登录业务处理：

1. 检查用户名是否存在
2. 检查用户名和密码是否正确

### 3.1.2 用户名验证

接口设计：

**/user/check\_valid.do** post

**request**

str , type

str可以是用户名也可以是email，对应的type是username和email

**Response**

**Fail**

{

“status” : 1 ,

“msg” : ”用户已存在”

}

**Success**

{

“status” : 0,

“msg” : ”校验成功”

}

### 3.1.3 注册

接口设计：

**/user/register.do** post

**request**

username ,password,email,phone,question,answer

**Response**

**Fail**

{

“status” : 1 ,

“msg” : ”用户已存在”

}

**Success**

{

“status” : 0,

“msg” : ”校验成功”

}

### 3.1.4 忘记密码问题获取

接口设计：

/user/forget\_get\_question.do

request

username

response

**Success**

{

“status” : 0,

“msg” : "这里是问题"

}

**Fail**

{

“status” : 1 ,

“msg” : ”该用户未设置找回密码问题”

}

Controller

1. 使用用户名查找问题

Service

1. 检查用户名是否存在
2. 查找问题并返回
3. 如果问题为空则返回“找回密码的问题是空的”：

Mapper

1、selectQuestionByUsername

### 3.1.5 提交问题答案

接口设计：

/user/forget\_check\_answer.do

request

Username,question,answer

response

正确的话返回值有一个token,修改密码的时候需要用到这个，传递给下一个接口，然后用guava来验证有效期

**Success**

{

“status” : 0,

“data” : "34a4c892-9663-2c3b-9a20-fb56839a76a5"

}

**Fail**

{

“status” : 1 ,

“msg” : ”问题答案错误”

}

Controller

1、返回service方法的值

Service

1. checkAnswer

2、生成forgetToken, UUID.randomUUID().toString()

3、调用TokenCache.setKey方法设置token，返回值传入forgetToken

Mapper

1. checkAnswer的方法，返回值为int ,如果返回值大于1则表示答案正确

创建TokenCache类

1. 声明日志
2. 声明静态内存块
3. 设置缓存的初始化容量为1000，最大为10000，当超过最大容量时，使用LRU算法移除缓存项，有效期为12个小时。

### 3.1.6 忘记密码的重置密码

接口设计：

/user/forget\_reset\_password.do

request

username,passwordNew,forgetToken

response

**Success**

{

“status” : 0,

“msg” : "修改密码成功"

}

**Fail**

{

“status” : 1 ,

“msg” : ”修改密码操作失败”

}

或

**Fail**

{

“status” : 1 ,

“msg” : ”token已经失效”

}

Controller

1. forgetResetPassword(username,passwordNew,forgetToken)

Service

1. 对 forgetToken 判空
2. 校验username
3. 获取token，并且校验是否为空
4. 判断传进来的forgetToken和从TokenCache中的token是否一致
5. 如果一致则更新密码

Mapper

1、resetPasswordByUsername

### 3.1.7 登录状态重置密码

接口设计：

/user/reset\_password.do

request

passwordOld,passwordNew

response

**Success**

{

“status” : 0,

“msg ” : "修改密码成功"

}

**Fail**

{

“status” : 1 ,

“msg” : ”旧密码输入错误”

}

Controller

1. 获取session中的用户信息
2. 判断是否为空，为空则为未登录
3. 调用service的重置密码方法

Service

1. 检查旧密码是否存在
2. 如果返回值为0则错误
3. 将新密码加密后传入到user中
4. 调用updateByPrimaryKeySelective(user)方法更新

Mapper

1. 根据旧密码和userId检查是否存在

### 3.1.8 获取用户信息

接口设计：

/user/get\_user\_info.do

request

无参数

response

**Success**

{

“status” : 0,

“data” : {

“id” : 12,

“username” : ”aaa”,

“email” : [”aaa@163.com”,](mailto:\”aaa@163.com\”,)

“phone”:null,

“role” : 0,

“createTime” : 1479048325000,

“updateTime” : 1479048325000

}

}

**Fail**

{

“status” : 1 ,

“msg” : ”用户未登陆，无法获取当前用户信息”

}

Controller

getUserInfo

1. 通过session获取当前用户信息，并强转为User类型
2. 判断如果user不为空返回 成功
3. 如果为空，返回用户未登录

### 3.1.9 登录状态更新个人信息

接口设计：

/user/update\_information.do

request

email,phone,question,answer

response

**Success**

{

“status” : 0,

“msg”:“更新个人信息成功”

}

**Fail**

{

“status” : 10 ,

“msg” : ”用户未登录,无法获取当前用户信息，status=10,强制登录”

}

Controller

1. 获取当前登录用户信息，并判断是否为空
2. 将当前的userId赋值给传进来的user

Service

1. username不能被更新
2. Email也要进行校验，校验新的email是否已经存在，并且存在的email如果相同的话，不能是我们当前的这个用户的
3. New一个新的User对象，并选择性地将user中的值传入
4. 更新用户信息

Mapper

1. checkEmailByUserId(String email ,Integer userId)

### 3.1.10 获取当前登录用户的详细信息，并强制登录

接口设计：

/user/get\_information.do

request

无参数

response

**Success**

{

“status” : 0,

“data” : {

“id” : 12,

“username” : ”aaa”,

“email” : [”aaa@163.com”,](mailto:\”aaa@163.com\”,)

“phone”:null,

“question” : ”question”,

“answer” : ”answer”,

“role” : 0,

“createTime” : 1479048325000,

“updateTime” : 1479048325000

}

}

**Fail**

{

“status” : 1 ,

“msg” : ”用户未登录,无法获取当前用户信息，status=10,强制登录”

}

Controller

1. 判断用户是否登录
2. 调用接口获取用户信息

Service

1. 调用 userMapper.selectByPrimaryKey(userId)来根据用户的信息
2. 如果返回值不为空则返回用户信息

Dao

1、selectByPrimaryKey

Mapper

1、

### 3.1.11 退出登录

接口设计：

/user/logout.do

request

无参数

通过session获取token

response

**Success**

{

“status” : 0,

“msg” : “登出成功"

}

**Fail**

{

“status” : 1 ,

“msg” : ”服务器异常”

}

Controller

1. 从session中删除当前用户的信息
2. 然后返回登出成功

## 3.2 后台接口

### 3.2.1 后台管理员登录

接口设计：

/manage/user/login.do

request

String username,

String password

**Success**

{

“status” : 0,

“data” : {

“id” : 12,

“username” : ”aaa”,

“email” : [”aaa@163.com”,](mailto:\”aaa@163.com\”,)

“phone”:null,

“role” : 0,

“createTime” : 1479048325000,

“updateTime” : 1479048325000

}

}

**Fail**

{

“status” : 1 ,

“msg” : ”密码错误”

}

Controller

1. 调用iUserService.login的方法登录
2. 然后检查登录获取到的角色是否是管理员角色，如果不是，则返回不是管理员
3. 如果是，则将用户信息写入session中
4. 最后返回登录的值

## 3.3 学习目标

### 3.3.1 横向越权纵向越权的安全漏洞

横向越权：攻击者尝试访问与他拥有相同权限的用户的资源

纵向越权：低级别攻击者访问高级别用户的资源

### 3.3.2 MD5明文加密及增加salt值

### 3.3.3 Guava缓存的使用

### 3.3.4 高复用服务响应对象的设计思想及抽象封装

### 3.3.5 Mybatis-plugin的使用技巧

### 3.3.6 Session的使用

### 3.3.7 方法局部演进

# 4 分类管理模块

## 4.1 分类管理模块开发概述

1、获取节点

2、增加节点

3、修改名字

4、获取分类ID

5、递归子节点ID

6、

## 4.2 学习目标

1、如何设计及封装无限层级的树状结构

2、递归算法的设计思想

3、如何处理复杂对象排重

4、重写hashcode和equal的注意事项

5、

## 4.3 数据库设计

parent\_id 为 0则为根节点



## 4.4 分类管理模块接口

### 4.4.1 获取品类子节点（平级）

/manage/category/get\_category.do

Request

categoryId(default = 0)

Response

Success

{

“status” : 0 ,

“data” : [

{

“id” : 2,

“parentId” : 1,

“name”:”手机”,

“status”:true,

“sortOrder” : 3,

“createTime” : 1479048325000,

“updateTime” : 1479048325000

},

{

“id” : 4,

“parentId” : 1,

“name”:”移动座机”,

“status”:true,

“sortOrder” : 5,

“createTime” : 1479048325000,

“updateTime” : 1479048325000

}

]

}

Fail

{

“status” : 10,

“msg” : “用户未登录，请登录”

}

或

{

“status”:1,

“msg”:”未找l到该品类”

}

Controller

1. 参数 session 、categoryId
2. 判断用户是否已经登录
3. 判断是否是管理员
4. 如果是则调用getChildrenParallelCategory 方法

Service

1. 参数categoryId
2. 调用mapper的查询方法
3. 判断返回值是否为空，为空则记录日志
4. 返回结果

Mapper

Dao

### 4.4.2 增加节点

/manage/category/add\_category.do

Request

parentId

categoryName

Success

{

“status” : 0,

“msg” : “增加品类成功”

}

Fail

{

“status”:1,

“msg”:”增加品类失败”

}

Controller

1. 参数 HttpSession session,String categoryName,String parentId 默认值为 0
2. 判断用户是否登录，如果没有登录则返回
3. 校验是否是管理员

Service

1. 判断父节点ID和品类名是否为空
2. 封装品类类
3. 插入数据库
4. 判断是否插入成功

Dao

1、

Mapper

### 4.4.3 修改品类名字

/manage/category/set\_category\_name.do

Request

categoryId

categoryName

Success

{

“status” : 0,

“msg” : “更新品类名字成功”

}

Fail

{

“status”:1,

“msg”:”更新品类名字失败”

}

Controller

1. 判断用户是否已经登录
2. 判断用户是否是管理员

Service

1. 参数Integer categoryId, String categoryName
2. 对参数判空
3. 封装Category类
4. 调用更新品类名的dao
5. 判断是否更新成功

Dao

1、

Mapper

### 4.4.4 获取当前分类ID以及递归子节点categoryId

/manage/category/get\_deep\_category.do

Request

categoryId

Success

{

“status” : 0,

“data” : {

100009,

100010,

100001,

100006,

100007,

100008

}

}

Fail

{

“status”:1,

“msg”:”更新品类名字失败”

}

Controller

1. 判断是否已经登录
2. 判断是否是管理员角色
3. 调用方法查询单前节点ID和递归子节点ID
4. 返回结果

Service

1. 递归算法算出单前节点和子节点的ID

Mapper

Dao

# 5 商品模块

## 5.1 功能介绍

### 5.1.1 前台功能

产品搜索

动态排序

商品详情

### 5.1.2 后台功能

商品列表

商品搜索

图片上传

富文本上传

商品详情

商品上下架

增加商品

更新商品

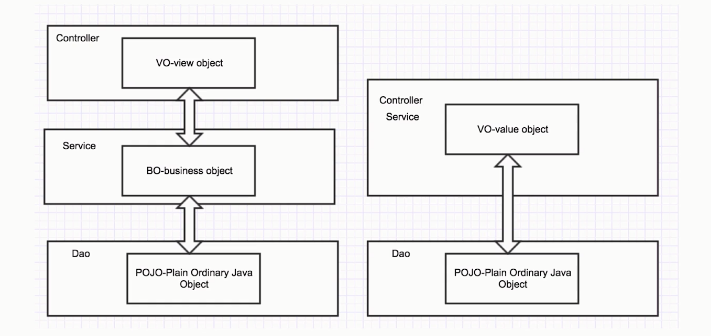
## 5.2 学习目标

### 5.2.1 FTP服务的对接

### 5.2.2 springMVC文件上传

### 5.2.3流读取Properties配置文件

### 5.2.4抽象POJO、BO、VO对象之间的转换关系及解决思路



### 5.2.5 joda-time快速入门

public static Date strToDate(String dateStr ,String dateFormatStr)  
{  
 DateTimeFormatter dateTimeFormatter = DateTimeFormat.*forPattern*(dateFormatStr);  
 DateTime dateTime = dateTimeFormatter.parseDateTime(dateStr);  
 return dateTime.toDate();  
}

### 5.2.6静态块

静态代码块在工程加载的时候就加载到JVM中，运行速度快

static {  
 String fileName = "mmall.properties";  
 *props* = new Properties();  
 try {  
 *props*.load(new InputStreamReader(PropertiesUtil.class.getClassLoader().getResourceAsStream(fileName),"UTF-8"));  
 } catch (IOException e) {  
 *logger*.error("配置文件读取异常",e);  
 }  
}

### 5.2.7 Mybatis-PageHelper高效准确地分页及动态排序

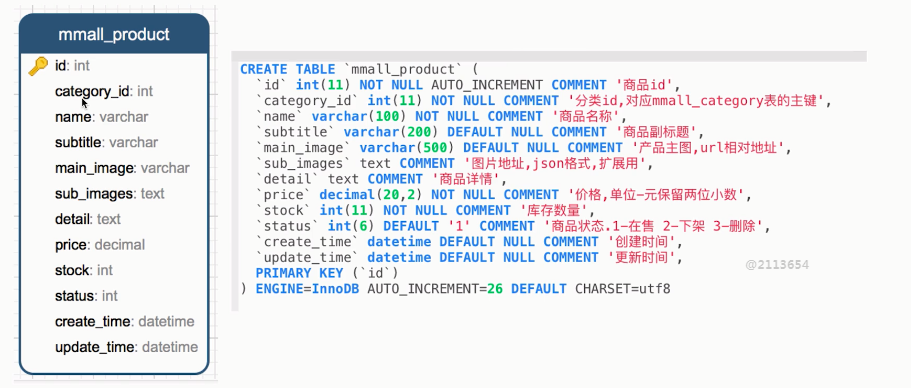
PageHelper

PageHelper.*startPage*(pageNum,pageSize);  
List<ProductListVo> productListVoList = Lists.*newArrayList*();  
List<Product> productList = productMapper.selectList();  
  
for(Product productItem : productList)  
{  
 ProductListVo productListVo = assembleProductListVo(productItem);  
 productListVoList.add(productListVo);  
}  
PageInfo pageResult = new PageInfo(productList);  
pageResult.setList(productListVoList);  
return ServerResponse.*createBySuccess*(pageResult);

### 5.2.7 Mybatis对List遍历的实现方法

### 5.2.7 Mybatis对where语句动态拼装的几个版本演变

## 5.3 数据表设计



## 5.4 接口设计

### 5.4.1 前台商品接口设计

#### 5.4.1.1 产品搜索及动态排序List

/product/list.do

Request

categoryId

keyword

pageNum(default=1)

pageSize(default=10)

orderBy(default=””) 排序参数，例如：price\_desc 、price\_asc

Response

{

"status": 0,

"data": {

"pageNum": 1,

"pageSize": 10,

"size": 2,

"orderBy": null,

"startRow": 1,

"endRow": 2,

"total": 2,

"pages": 1,

"list": [

{

"id": 1,

"categoryId": 3,

"name": "iphone7",

"subtitle": "双十一促销",

"mainImage": "mainimage.jpg",

"status":1,

"price": 7199.22

},

{

"id": 2,

"categoryId": 2,

"name": "oppo R8",

"subtitle": "oppo促销进行中",

"mainImage": "mainimage.jpg",

"status":1,

"price": 2999.11

}

],

"firstPage": 1,

"prePage": 0,

"nextPage": 0,

"lastPage": 1,

"isFirstPage": true,

"isLastPage": true,

"hasPreviousPage": false,

"hasNextPage": false,

"navigatePages": 8,

"navigatepageNums": [

1

]

}

}

Controller

list

Service

Dao

Mapper

#### 5.4.1.2 产品detail

/product/detail.do

Request

productId

Response

Success

{

"status": 0,

"data": {

"id": 2,

"categoryId": 2,

"name": "oppo R8",

"subtitle": "oppo促销进行中",

"mainImage": "mainimage.jpg",

"subImages": "[\"mmall/aa.jpg\",\"mmall/bb.jpg\",\"mmall/cc.jpg\",\"mmall/dd.jpg\",\"mmall/ee.jpg\"]",

"detail": "richtext",

"price": 2999.11,

"stock": 71,

"status": 1,

"createTime": "2016-11-20 14:21:53",

"updateTime": "2016-11-20 14:21:53"

}

}

Fail

{

"status": 1,

"msg": "该商品已下架或删除"

}

### 5.4.2 后台商品接口设计

#### 5.4.2.1 产品List

/manage/product/list.do

Request

pageNum(defaultValue =1)

pageSize(defaultValue =10)

Response

Success

{

"status": 0,

"data": {

"pageNum": 1,

"pageSize": 10,

"size": 2,

"orderBy": null,

"startRow": 1,

"endRow": 2,

"total": 2,

"pages": 1,

"list": [

{

"id": 1,

"categoryId": 3,

"name": "iphone7",

"subtitle": "双十一促销",

"mainImage": "mainimage.jpg",

"status":1,

"price": 7199.22

},

{

"id": 2,

"categoryId": 2,

"name": "oppo R8",

"subtitle": "oppo促销进行中",

"mainImage": "mainimage.jpg",

"status":1,

"price": 2999.11

}

],

"firstPage": 1,

"prePage": 0,

"nextPage": 0,

"lastPage": 1,

"isFirstPage": true,

"isLastPage": true,

"hasPreviousPage": false,

"hasNextPage": false,

"navigatePages": 8,

"navigatepageNums": [

1

]

}

}

Fail

{

"status": 10,

"msg": "用户未登录,请登录"

}

Servce

getProductList

//startPage--start

//填充自己的sql查询逻辑

//pageHelper--收尾

Dao

#### 5.4.2.2 产品搜索

/manage/product/search.do

Request

productName

productId

pageNum(default=1)

pageSize(default=10)

Response

Success

{

"status": 0,

"data": {

"pageNum": 1,

"pageSize": 10,

"size": 1,

"orderBy": null,

"startRow": 1,

"endRow": 1,

"total": 1,

"pages": 1,

"list": [

{

"id": 1,

"categoryId": 3,

"name": "iphone7",

"subtitle": "双十一促销",

"mainImage": "mainimage.jpg",

"price": 7199.22

}

],

"firstPage": 1,

"prePage": 0,

"nextPage": 0,

"lastPage": 1,

"isFirstPage": true,

"isLastPage": true,

"hasPreviousPage": false,

"hasNextPage": false,

"navigatePages": 8,

"navigatepageNums": [

1

]

}

}

Fail

{

"status": 10,

"msg": "用户未登录,请登录"

}

Controller

Service

Dao

#### 5.4.2.3 图片上传

/manage/product/update.do

**Request**

<form name="form2" action="/manage/product/upload.do" method="post" enctype="multipart/form-data">

<input type="file" name="upload\_file">

<input type="submit" value="upload"/>

</form>

**Response**

**Success**

{

"status": 0,

"data": {

"uri": "e6604558-c0ff-41b9-b6e1-30787a1e3412.jpg",

"url": "http://img.happymmall.com/e6604558-c0ff-41b9-b6e1-30787a1e3412.jpg"

}

}

**Fail**

status!=0的时候

#### 5.4.2.4 状态设置

/manage/product/set\_sale\_status.do

**Request**

productId

status

**Response**

**Success**

{

"status": 0,

"msg": "产品状态更新成功"

}

**Fail**

{

"status": 0,

"msg": "产品状态更新失败"

}

**Controller**

1. 方法名 setProductStatus,参数session,productId,status
2. 获取当前登录用户，并判断用户是否已经登录
3. 判断用户是否管理员

**Service**

1. setSaleStatus参数 productId ,status
2. 判断是否为空，如果为空则直接返回非法参数
3. 封装Product对象
4. 调用方法更新产品
5. 判断返回值，并根据返回值返回结果

#### 5.4.2.5 产品详情

/manage/product/detail.do

**Request**

productId

**Response**

**Success**

{

"status": 0,

"data": {

"id": 2,

"categoryId": 2,

"parentCategoryId":1,

"name": "oppo R8",

"subtitle": "oppo促销进行中",

"imageHost": "http://img.happymmall.com/",

"mainImage": "mainimage.jpg",

"subImages": "[\"mmall/aa.jpg\",\"mmall/bb.jpg\",\"mmall/cc.jpg\",\"mmall/dd.jpg\",\"mmall/ee.jpg\"]",

"detail": "richtext",

"price": 2999.11,

"stock": 71,

"status": 1,

"createTime": "2016-11-20 14:21:53",

"updateTime": "2016-11-20 14:21:53"

}

}

**Fail**

{

"status": 1,

"msg": "没有权限"

}

Controller

1. 方法名 getDetail,参数session,productId
2. 获取当前登录用户，并判断用户是否已经登录
3. 判断用户是否管理员

Service

1. manageProductDetail,参数是productId
2. 判断如果参数为空，则返回参数错误
3. 根据productId获取商品信息
4. 如果返回对象为空，则返回“产品已经下架”

VO

1. 创建一个ProductDetailVo类
2. Id、productId、name、subtitle、mainImage、subImage、
3. detail、price、stock库存、status、createTime、imageHost

#### 5.4.2.6 新增或更新产品

/manage/product/save.do

**Request**

categoryId

name

subtitle

mainImage

subImage

detail

price

status

stock

Id

**Response**

**success**

{

"status": 0,

"data": "更新产品成功"

}

或

{

"status": 0,

"data": "新增产品成功"

}

**fail**

{

"status": 1,

"data": "更新产品失败"

}

Controller

1. 方法名 productSave,参数session,product
2. 获取当前登录用户，并判断用户是否已经登录
3. 判断用户是否管理员
4. 调用保存产品的方法

Service

1. 判断产品类是否为空，不为空则继续，为空则直接返回新增或更新产品参数不正确
2. 判断产品类的子照片是否不为空
3. 通过数组将图片存在数组中
4. 将首张照片赋值给主照片
5. 判断product.getId是否为空
6. 不为空则调用方法更新
7. 如果为空则新增产品

#### 5.4.2.7 富文本上传图片

/manage/product/richtext\_img\_upload.do

**Request**

file

Upload

<form name="form2" action="/manage/product/upload.do" method="post" enctype="multipart/form-data">

<input type="file" name="upload\_file">

<input type="submit" value="upload"/>

</form>

**Response**

**Success**

{

“file\_path”:””,

“msg”:”上传成功”,

”success”: true

}

Service

1. 获取文件名
2. 获取扩展名
3. 生成新的文件名
4. 新建文件夹对象，判断文件夹是否存在，如果不存在，则赋予可写权限，因为tomcat下没有创建文件夹的权限，必须要赋予权限。
5. 创建dir
6. 创建目标文件
7. 上传文件到本地文件夹
8. 将本地的文件上传至服务器
9. 捕获异常
10. 返回文件名 targetFile.getName()

FTPUtil

1. 获取IP、user、pass
2. 初始化
3. 公共静态方法 uploadFile
4. 私有方法uploadFile，判断连接是否成功，修改工作目录，设置缓存区1024，设置编码为UTF-8，设置文件类型为二进制FTPClient.BINARY\_FILE\_TYPE,设置本地被动模式enterLocalPassiveMode()
5. For循环上传文件
6. 连接服务器的方法，先创建FTPClient对象，连接ip，使用login方法登录

# 6 购物车模块

## 6.1 功能介绍

加入商品

更新商品数

查询商品数

移除商品

单选/取消

全选/取消

购物车列表

### 6.1.1 前台功能

## 6.2 学习目标

### 6.2.1购物车模块设计思想

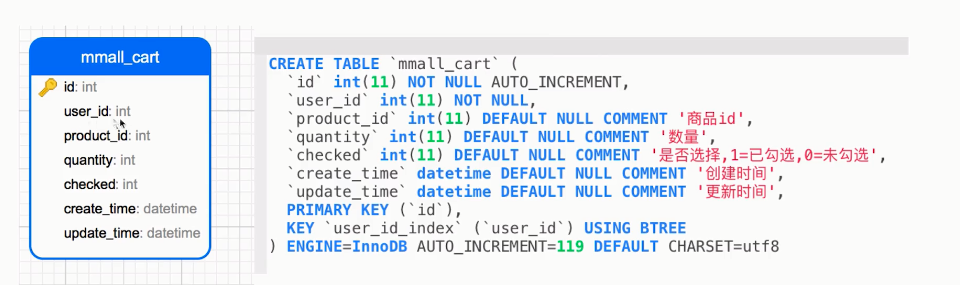
### 6.2.2如何封装一个高复用的购物车核心方法

### 6.2.3解决浮点型商业运算中丢失精度问题

//加  
public static BigDecimal add(Double v1 ,Double v2)  
{  
 BigDecimal b1 = new BigDecimal(v1.toString());  
 BigDecimal b2 = new BigDecimal(v2.toString());  
 return b1.add(b2);  
}  
//减  
public static BigDecimal sub(Double v1 ,Double v2)  
{  
 BigDecimal b1 = new BigDecimal(v1.toString());  
 BigDecimal b2 = new BigDecimal(v2.toString());  
 return b1.subtract(b2);  
}  
  
//乘  
public static BigDecimal mul(Double v1 ,Double v2)  
{  
 BigDecimal b1 = new BigDecimal(v1.toString());  
 BigDecimal b2 = new BigDecimal(v2.toString());  
 return b1.multiply(b2);  
}  
  
//除,需要保留两位小数点，并且四舍五入  
public static BigDecimal div(Double v1 ,Double v2)  
{  
 BigDecimal b1 = new BigDecimal(v1.toString());  
 BigDecimal b2 = new BigDecimal(v2.toString());  
 return b1.divide(b2,2,BigDecimal.*ROUND\_HALF\_UP*);  
}

### 6.2.4解决浮点型商业运算中丢失精度问题

## 6.3 数据表设计



## 6.4 接口设计

### 6.4.1 前台购物车接口设计

#### 6.4.1.1 购物车List列表

/cart/list.do

http://localhost:8089/[/cart/list.do](http://localhost:8080/cart/list.do)

注意点：

需要先登录,所有的密码都是123

NEED\_LOGIN(10, "NEED\_LOGIN"),//需要登录的错误编码

价格的单位是元,保留小数后2位

Request

无参数，需要登录状态

Response

success

{

"status": 0,

"data": {

"cartProductVoList": [

{

"id": 1,

"userId": 13,

"productId": 1,

"quantity": 1,

"productName": "iphone7",

"productSubtitle": "双十一促销",

"productMainImage": "mainimage.jpg",

"productPrice": 7199.22,

"productStatus": 1,

"productTotalPrice": 7199.22,

"productStock": 86,

"productChecked": 1,

"limitQuantity": "LIMIT\_NUM\_SUCCESS"

},

{

"id": 2,

"userId": 13,

"productId": 2,

"quantity": 1,

"productName": "oppo R8",

"productSubtitle": "oppo促销进行中",

"productMainImage": "mainimage.jpg",

"productPrice": 2999.11,

"productStatus": 1,

"productTotalPrice": 2999.11,

"productStock": 86,

"productChecked": 1,

"limitQuantity": "LIMIT\_NUM\_SUCCESS"

}

],

"allChecked": true,

"cartTotalPrice": 10198.33

}

}

fail

{

"status": 10,

"msg": "用户未登录,请登录"

}

#### 6.4.1.2 购物车添加商品

****/cart/add.do****

http://localhost:8089/[/cart/add.do](http://localhost:8080/cart/list.do)

注意点：

请注意这个字段，超过数量会返回这样的标识"limitQuantity"

失败的：LIMIT\_NUM\_FAIL

成功的：LIMIT\_NUM\_SUCCESS

Request

productId

count

Response

Success

{

"status": 0,

"data": {

"cartProductVoList": [

{

"id": 1,

"userId": 13,

"productId": 1,

"quantity": 12,

"productName": "iphone7",

"productSubtitle": "双十一促销",

"productMainImage": "mainimage.jpg",

"productPrice": 7199.22,

"productStatus": 1,

"productTotalPrice": 86390.64,

"productStock": 86,

"productChecked": 1,

"limitQuantity": "LIMIT\_NUM\_SUCCESS"

},

{

"id": 2,

"userId": 13,

"productId": 2,

"quantity": 1,

"productName": "oppo R8",

"productSubtitle": "oppo促销进行中",

"productMainImage": "mainimage.jpg",

"productPrice": 2999.11,

"productStatus": 1,

"productTotalPrice": 2999.11,

"productStock": 86,

"productChecked": 1,

"limitQuantity": "LIMIT\_NUM\_SUCCESS"

}

],

"allChecked": true,

"cartTotalPrice": 89389.75

}

}

fail

{

"status": 10,

"msg": "用户未登录,请登录"

}

#### 6.4.1.3更新购物车某个商品数量

****/cart/update.do****

http://localhost:8089/[/cart/update.do](http://localhost:8080/cart/list.do)

Request

productId

count

Response

{

"status": 0,

"data": {

"cartProductVoList": [

{

"id": 1,

"userId": 13,

"productId": 1,

"quantity": 12,

"productName": "iphone7",

"productSubtitle": "双十一促销",

"productMainImage": "mainimage.jpg",

"productPrice": 7199.22,

"productStatus": 1,

"productTotalPrice": 86390.64,

"productStock": 86,

"productChecked": 1,

"limitQuantity": "LIMIT\_NUM\_SUCCESS"

},

{

"id": 2,

"userId": 13,

"productId": 2,

"quantity": 1,

"productName": "oppo R8",

"productSubtitle": "oppo促销进行中",

"productMainImage": "mainimage.jpg",

"productPrice": 2999.11,

"productStatus": 1,

"productTotalPrice": 2999.11,

"productStock": 86,

"productChecked": 1,

"limitQuantity": "LIMIT\_NUM\_SUCCESS"

}

],

"allChecked": true,

"cartTotalPrice": 89389.75

}

}

fail

{

"status": 10,

"msg": "用户未登录,请登录"

}

#### 6.4.1.4移除购物车某个商品

/cart/delete\_product.do

[http://localhost:8089/cart/delete\_product.do?productIds=1,3](http://localhost:8080/cart/delete_product.do?productIds=1,3)

Request

productIds

Response

{

"status": 0,

"data": {

"cartProductVoList": [

{

"id": 2,

"userId": 13,

"productId": 2,

"quantity": 1,

"productName": "oppo R8",

"productSubtitle": "oppo促销进行中",

"productMainImage": "mainimage.jpg",

"productPrice": 2999.11,

"productStatus": 1,

"productTotalPrice": 2999.11,

"productStock": 86,

"productChecked": 1,

"limitQuantity": "LIMIT\_NUM\_SUCCESS"

}

],

"allChecked": true,

"cartTotalPrice": 2999.11

}

}

fail

{

"status": 10,

"msg": "用户未登录,请登录"

}

#### 6.4.1.5购物车选中某个商品

/cart/select.do

[http://localhost:8089/cart/select.do?productId=1](http://localhost:8080/cart/delete_product.do?productIds=1,3)

Request

productId

Response

{

"status": 0,

"data": {

"cartProductVoList": [

{

"id": 2,

"userId": 13,

"productId": 2,

"quantity": 1,

"productName": "oppo R8",

"productSubtitle": "oppo促销进行中",

"productMainImage": "mainimage.jpg",

"productPrice": 2999.11,

"productStatus": 1,

"productTotalPrice": 2999.11,

"productStock": 86,

"productChecked": 1,

"limitQuantity": "LIMIT\_NUM\_SUCCESS"

}

],

"allChecked": true,

"cartTotalPrice": 2999.11

}

}

fail

{

"status": 10,

"msg": "用户未登录,请登录"

}

#### 6.4.1.6购物车取消选中某个商品

/cart/un\_select.do

[http://localhost:8089/cart/un\_select.do?productId=2](http://localhost:8080/cart/delete_product.do?productIds=1,3)

Request

productId

Response

success

{

"status": 0,

"data": {

"cartProductVoList": [

{

"id": 2,

"userId": 13,

"productId": 2,

"quantity": 1,

"productName": "oppo R8",

"productSubtitle": "oppo促销进行中",

"productMainImage": "mainimage.jpg",

"productPrice": 2999.11,

"productStatus": 1,

"productTotalPrice": 2999.11,

"productStock": 86,

"productChecked": 0,

"limitQuantity": "LIMIT\_NUM\_SUCCESS"

}

],

"allChecked": true,

"cartTotalPrice": 0

}

}

fail

{

"status": 10,

"msg": "用户未登录,请登录"

}

#### 6.4.1.7查询在购物车里的产品数量

/cart/get\_cart\_producnt\_count.do

[http://localhost:8089/get\_cart\_producnt\_count.do?productId=2](http://localhost:8080/cart/delete_product.do?productIds=1,3)

Request

无

Response

{

"status": 0,

"data": 0

}

fail

{

"status": 10,

"msg": "出现异常"

}

#### 6.4.1.8购物车全选

/cart/select\_all.do

[http://localhost:8089/select\_all.do](http://localhost:8080/cart/delete_product.do?productIds=1,3)

注意返回值中的cartTotalPrice的变化

Request

Response

success

{

"status": 0,

"data": {

"cartProductVoList": [

{

"id": 2,

"userId": 13,

"productId": 2,

"quantity": 1,

"productName": "oppo R8",

"productSubtitle": "oppo促销进行中",

"productMainImage": "mainimage.jpg",

"productPrice": 2999.11,

"productStatus": 1,

"productTotalPrice": 2999.11,

"productStock": 86,

"productChecked": 0,

"limitQuantity": "LIMIT\_NUM\_SUCCESS"

}

],

"allChecked": true,

"cartTotalPrice": 2999.11

}

}

fail

{

"status": 10,

"msg": "用户未登录,请登录"

}

#### 6.4.1.9购物车取消全选

/cart/un\_select\_all.do

[http://localhost:8089/un\_select\_all.do](http://localhost:8080/cart/delete_product.do?productIds=1,3)

注意返回值中的cartTotalPrice的变化

Request

Response

success

{

"status": 0,

"data": {

"cartProductVoList": [

{

"id": 2,

"userId": 13,

"productId": 2,

"quantity": 1,

"productName": "oppo R8",

"productSubtitle": "oppo促销进行中",

"productMainImage": "mainimage.jpg",

"productPrice": 2999.11,

"productStatus": 1,

"productTotalPrice": 2999.11,

"productStock": 86,

"productChecked": 0,

"limitQuantity": "LIMIT\_NUM\_SUCCESS"

}

],

"allChecked": true,

"cartTotalPrice": 0

}

}

fail

{

"status": 10,

"msg": "用户未登录,请登录"

}

# 7 常见错误