

# 慕课网Spring boot电商秒杀项目实战

**项目背景和搭建**

**框架的分层设计**

**项目功能模块**

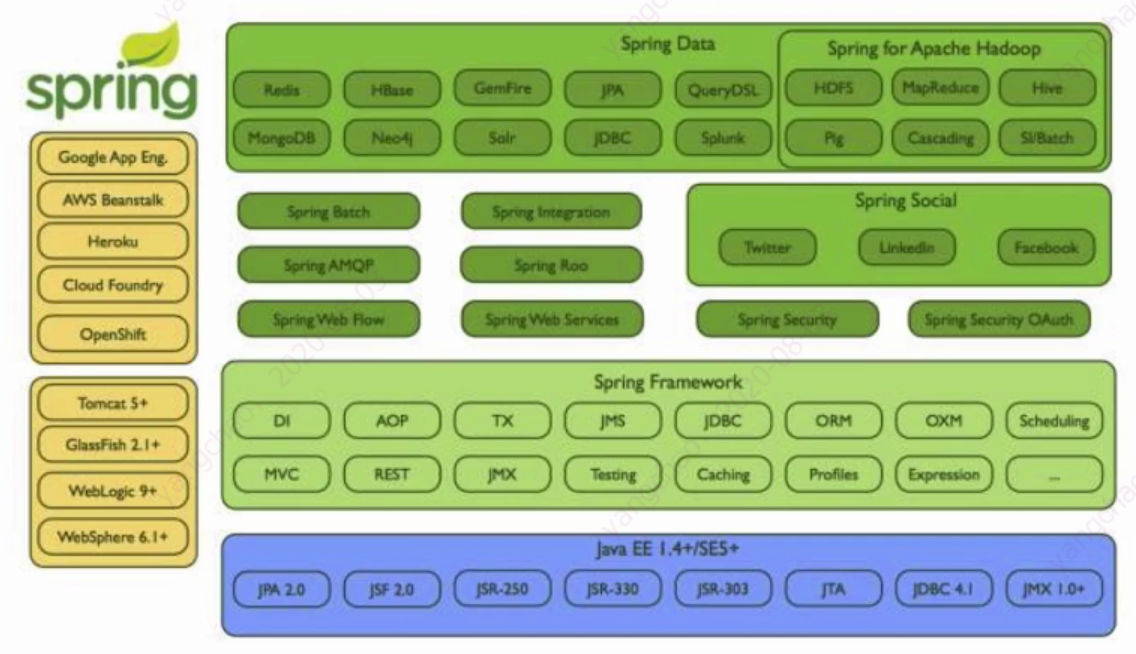
**数据库表格的关联设计**

**学习中遇到的问题及解决方案**

**项目的重难点**

**后续需要深入学习的技术点**

Spring框架



Spring SpringMVC Springboot三者的关系：

<https://zhuanlan.zhihu.com/p/63117304>

# 项目背景和搭建：

电商秒杀应用背景简介：

1. 用户通过手机号注册和登录
2. 进入商品详情页获取商品详情
3. 秒杀开始后进入下单页下单支付成功

工具环境：

使用IDEA+Maven搭建Springboot开发环境

集成Mybatis操作MySQL数据库

搭建项目：

Project->Maven-> maven-archetype-quickstart

Springboot 依赖和Mybatis依赖：

<https://www.cnblogs.com/victorbu/p/10538615.html>

[IDEA 中使用 MyBatis-generator 自动生成 MyBatis 代码](https://segmentfault.com/a/1190000009058867)

<https://segmentfault.com/a/1190000009058867>

<https://www.jianshu.com/p/b519e9ef605f>

<https://blog.csdn.net/awecoder/article/details/101165599>

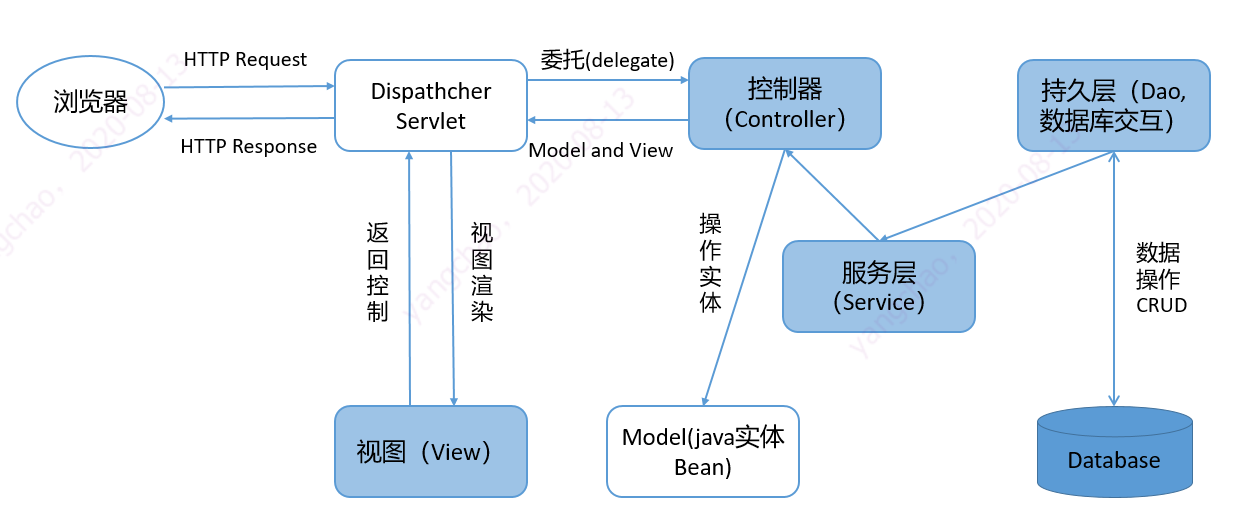
创建mybatis-generator.xml，根据数据库表格字段自动生成

<?xml version="1.0" encoding="UTF-8"?>  
<!DOCTYPE generatorConfiguration  
 PUBLIC "-//mybatis.org//DTD MyBatis Generator Configuration 1.0//EN"  
 "http://mybatis.org/dtd/mybatis-generator-config\_1\_0.dtd">  
  
<!--http://www.mybatis.org/generator/configreference/xmlconfig.html-->  
  
<generatorConfiguration>  
 <!-- 指定数据连接驱动jar地址 -->  
 <classPathEntry location="C:/Users/yangchao/.m2/repository/mysql/mysql-connector-java/8.0.21/mysql-connector-java-8.0.21.jar"/>  
 <context id="DB2Tables" targetRuntime="MyBatis3">  
 <jdbcConnection driverClass="com.mysql.cj.jdbc.Driver"  
 connectionURL="jdbc:mysql://127.0.0.1:3306/miaosha?serverTimezone=Asia/Shanghai"  
 userId="root"  
 password="password">  
 </jdbcConnection>  
  
 <!--生成 DataObject 类存放位置-->  
 <javaModelGenerator targetPackage="com.miaoshaproject.dataobject" targetProject="src/main/java">  
 <property name="enableSubPackages" value="true" />  
 <property name="trimStrings" value="true" />  
 </javaModelGenerator>  
  
 <!--生成映射文件存放位置-->  
 <sqlMapGenerator targetPackage="mapping" targetProject="src/main/resources">  
 <property name="enableSubPackages" value="true" />  
 </sqlMapGenerator>  
  
 <!--生成 Dao 类存放位置-->  
 <!--客户端代码，生成易于使用的针对 Model 对象和 XML 配置文件的代码-->  
 <!--type="ANNOTATEDMAPPER", 生成 Java Model 和基于注解的 Mapper 对象-->  
 <!--type="MIXEDMAPPER", 生成基于注解的 Java Model 和相应的 Mapper 对象-->  
 <!--type="XMLMAPPER", 生成 SQLMap XML 文件和独立的 Mapper 接口-->  
 <javaClientGenerator type="XMLMAPPER" targetPackage="com.miaoshaproject.dao" targetProject="src/main/java">  
 <property name="enableSubPackages" value="true" />  
 </javaClientGenerator>  
  
 <!--生成对应表及类名-->  
 <!--\*Example 禁用自动生成的复杂查询-->  
 <table tableName="user\_info" domainObjectName="UserDO" enableCountByExample="false" enableUpdateByExample="false"  
 enableDeleteByExample="false" enableSelectByExample="false"  
 selectByExampleQueryId="false"></table>  
 <table tableName="user\_password" domainObjectName="UserPasswordDO" enableCountByExample="false" enableUpdateByExample="false"  
 enableDeleteByExample="false" enableSelectByExample="false"  
 selectByExampleQueryId="false"></table>  
  
 </context>  
</generatorConfiguration>

生成的文件：ItemDO.java、 IItemDOMapper.java ItemDOMapper.xml

# 框架分层设计：

Spring MVC 模型设计：



1.View层：表示层，负责前端页面表示。

2.Controller层：负责具体业务模块流程的控制，即调用Service层的接口来控制业务流程。负责url映射（action）。

3.Service层：建立在DAO层之上，Controller层之下。调用Dao层的接口（Mapper.java），为Controller层提供接口。负责业务模块的逻辑应用设计，首先设计接口，再设计其实现的类。

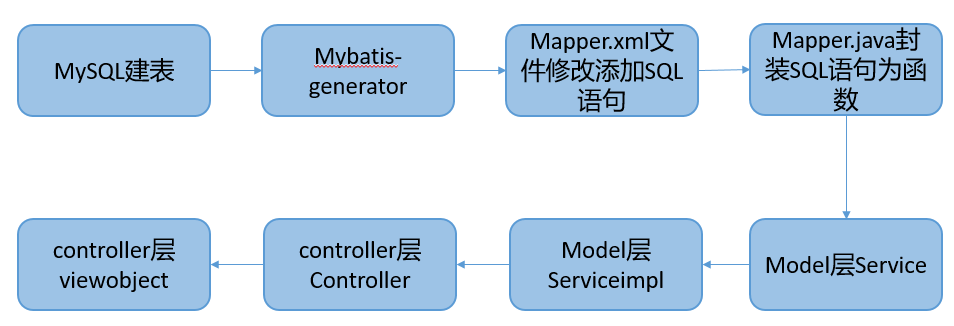
4.Dao层：负责数据持久化，与数据库进行联络的任务都封装在其中，Dao层的数据源以及相关的数据库连接参数都在Spring配置文件中进行配置。Dao接口中的方法都大同小异，因为对数据库的基本操作类似：insert、delete、update，select。

项目中分层和调用关系：

**controller层**： 与前端页面UI交互的层  
**controller->viewobject层**：视图，封装了向前端展示的数据，避免领域模型的某些敏感如用户密码等数据直接传给前端。  
**dao层**：与底层数据库交互的层，进行数据库的增删改查。  
**dao->dataobject层**：数据库表对于Java实体类的映射，是数据库表中数据的封装。  
**service层**：处于上层controller与下层dao层的中间层，处理各种逻辑业务和服务请求。

**service->model层**：领域模型，将不同的表但属于一个领域的字段的整合。

**service->impl层**：领域模型，model字段在业务功能中的具体应用与实现。  
**error层**：统一错误的格式，拦截tomcat不能处理的各类异常与错误。  
**response层**：返回给前端统一的格式(response+data)。

****

# 项目功能模块：

用户注册登录:

用户通过手机号注册，通过手机号和密码登录

商品模块：

用户可以添加商品，浏览商品列表，设置秒杀活动商品列表

交易下单模块：

在商品列表点击任意商品，进入商品详情或者下单

下单时，校验下单商品是否存在，用户是否登录，校验下单商品数量，

校验商品活动信息

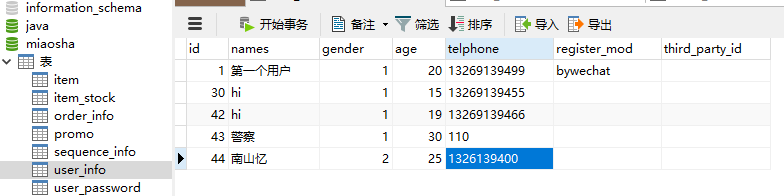
下单成功，订单信息入库，生成订单流水号，库存减一，销量加一

# 数据库表格的关联设计：

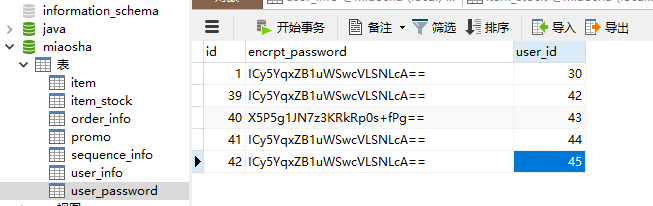
用户信息与用户密码表格的设计：

为了保护用户密码信息不被泄露，分为两个表格设计：

**User\_info:**



**User\_password: 用户密码加密后存放在数据库中**



user\_id=id 登录时用telphone校验对应的密码是否存在，以及用户输入密码和数据库中的密码是否相同。

用户信息就是用户本身，用户名和密码只是登陆钥匙

性能方面考虑

登陆验证的时候列较少，查询速度快。

安全性考虑

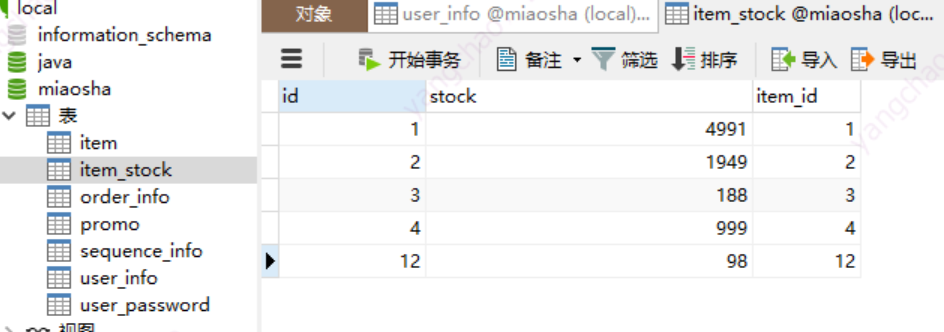
防止在查询用户信息时，把密码也直接查询出来，会容易被攻击和进行恶意操作

商品数据表设置：

商品信息：item:

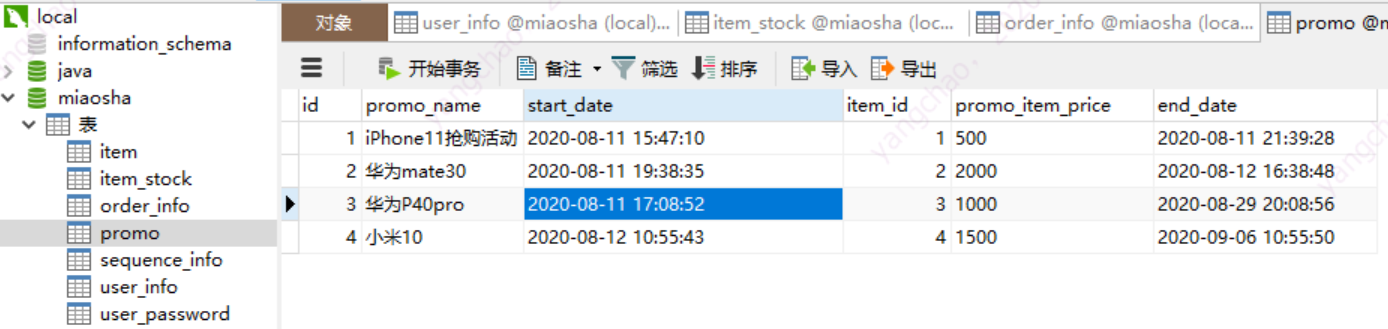


商品库存信息：



item\_id绑定item表中的id，

商品活动信息：

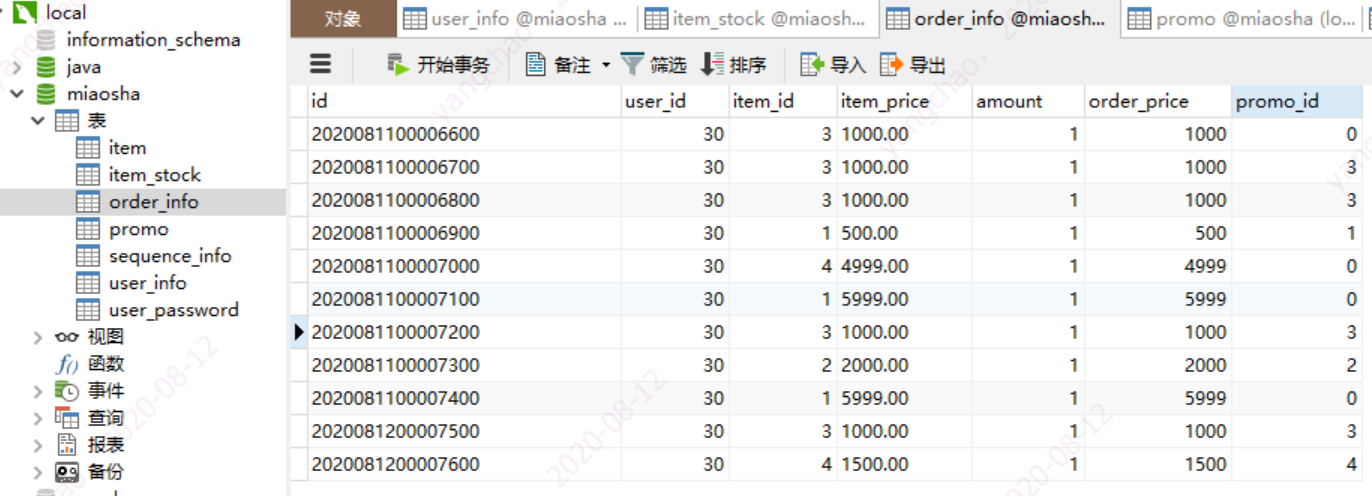


商品活动信息，item\_id绑定 item表格中商品id，人为设置活动的起始和结束时间，活动价格。

//秒杀活动状态 1:未开始 2：进行中 3：活动已结束  
private Integer status;

//判断当前时间是否秒杀活动即将开始或正在进行中  
  
if(promoModel.getStartDate().isAfterNow()){  
 promoModel.setStatus(1);  
}else if(promoModel.getEndDate().isBeforeNow()){  
 promoModel.setStatus(3);  
}else {  
 promoModel.setStatus(2);  
}

**订单信息：**



Id为订单号，前8位为时间，中间6位为自增序列，最后两位为分库分页表，项目中默认为为00.

Promo\_id: 为0表示为该商品没有秒杀活动，为平销价格，非零时表示该商品在商品活动列表中的id

Item\_price 为下单价格，如果商品在活动列表中且正在活动中，下单价格为活动价格，如果商品活动还未开始，不能下单，如果商品不在活动列表中，以平销价格下单。

**商品信息和下单信息的关联：下单成功，库存item\_stock 减一，item\_sales销量加一**

@Override  
@Transactional  
public boolean decreaseStock(Integer itemId, Integer amount) throws BusinessException {  
 int affectedRow=itemStockDOMapper.decreaseStock(itemId,amount);  
 if(affectedRow>0){  
 //库存更新成功  
 return true;  
 }else {  
 return false;  
 }  
}  
  
@Override  
@Transactional  
public void increaseSales(Integer itemId, Integer amount) throws BusinessException {  
 itemDOMapper.increaseSales(itemId,amount);  
}

<update id="increaseSales" >  
 update item  
 set sales = sales+ #{amount}  
 where id = #{id,jdbcType=INTEGER}  
</update>

<update id="decreaseStock" >  
 update item\_stock  
 set stock = stock-#{amount}  
 where item\_id = #{itemId} and stock >=#{amount}  
</update>

# 学习过程中遇到的问题及解决方案：

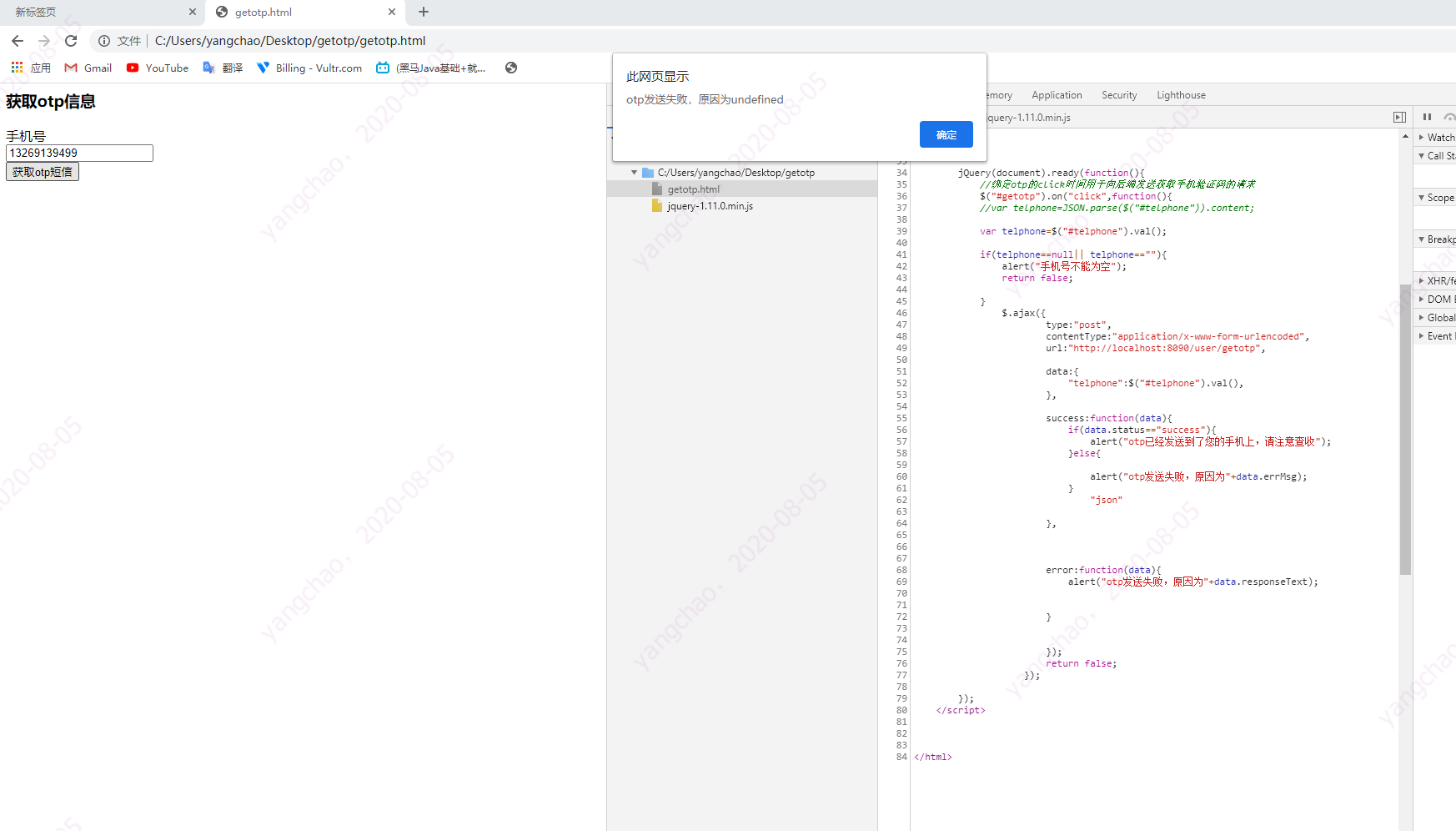
IDEA 报错 Could not autowire. No beans of 'UserMapper' type found

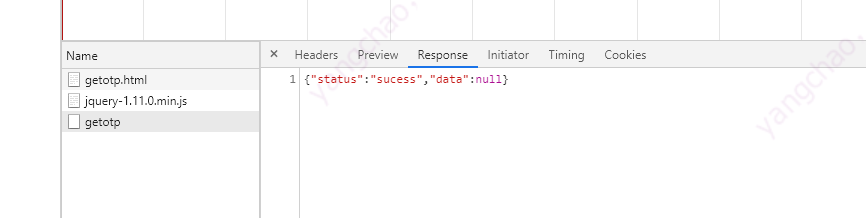
<https://blog.csdn.net/liuyinfei_java/article/details/79750475>

在显示SSL验证提示时:CST 2017 WARN: Establishing SSL connection without server's identity verificatio

<https://blog.csdn.net/dianyanxia/article/details/84026059>

**前端获取手机号错误：**





问题：CommonReturnType.java中success字段书写错误为sucess:

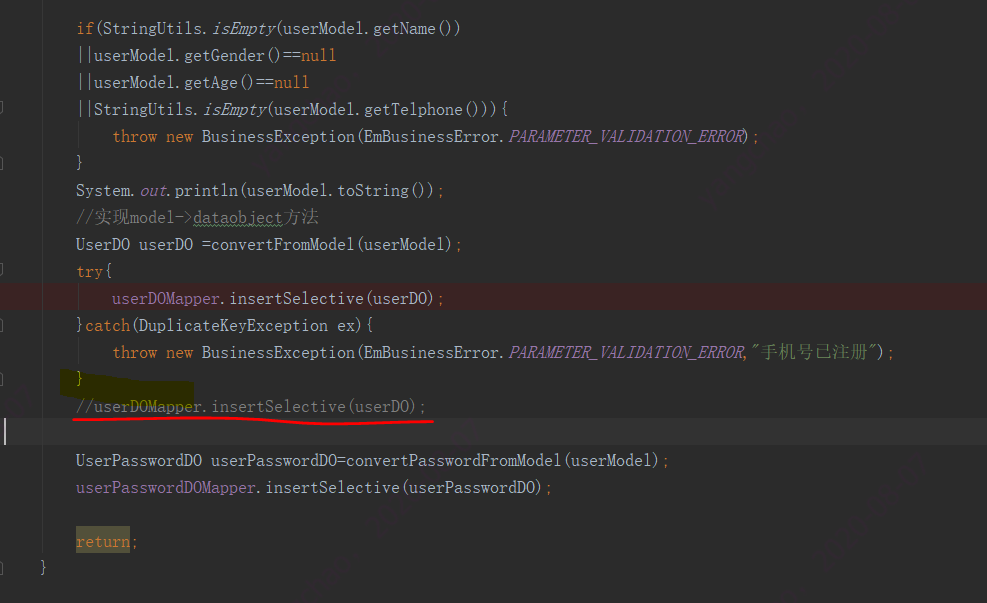
//定义一个通用的创建方法  
public static CommonReturnType create(Object result){  
 return CommonReturnType.*create*(result,"success");  
}

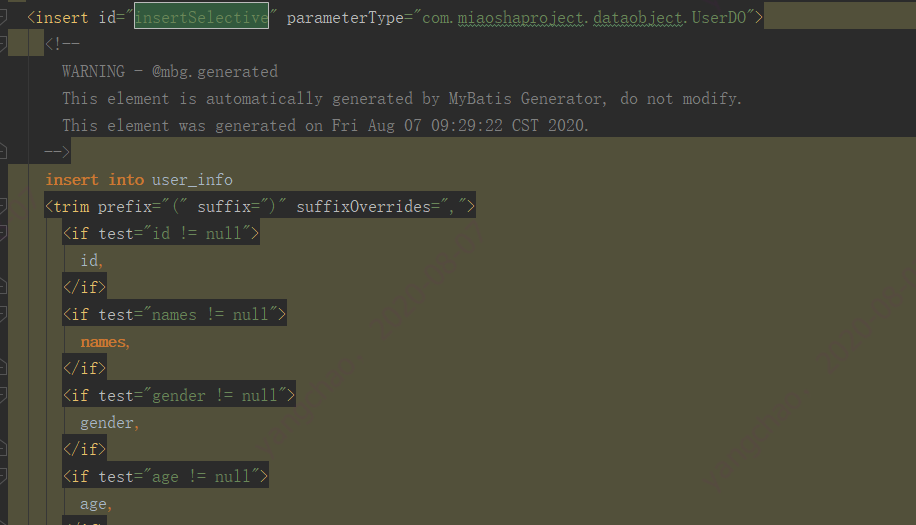
前端需要细心！！！

*用户注册时，MySQL写入错误*

SQL: insert into user\_info ( id, name, gender, age, telphone ) values ( ?, ?, ?, ?, ? )

Cause: java.sql.SQLIntegrityConstraintViolationException: Duplicate entry '5' for key 'PRIMARY'





第一次id默认为null,数据库自增处理，第二次用的是第一次的id,不是null,和上次一样了

MySQL主键自增获取到为null的得问题

解决方式：mybatis-generator.xml 添加

<generatedKey column="id" sqlStatement="Mysql" identity="true"/>

这样做的意义是，调用mapper插入一条数据到数据库之后，能从当前这个Entity中获取到插入之后的Id

<table tableName="user\_info" domainObjectName="UserDO" enableCountByExample="false" enableUpdateByExample="false"  
 enableDeleteByExample="false" enableSelectByExample="false"  
 selectByExampleQueryId="false">  
 <generatedKey column="id" sqlStatement="Mysql" identity="true"/>  
</table>

参考：<https://blog.csdn.net/u013282737/article/details/89491187>

UserPasswordDO userPasswordDO=convertPasswordFromModel(userModel);  
userPasswordDO.setUserId(userDO.getId()); //通过user\_id=id 将user\_info和user\_password关联起来  
userPasswordDOMapper.insertSelective(userPasswordDO);

**商品信息写入数据库失败：**

Error:(47, 51) java: 无法从静态上下文中引用非静态 方法 createItem(com.miaoshaproject.service.model.ItemModel)

//创建商品的controller  
  
//@RequestMapping(value = "/create",method={RequestMethod.POST},consumes = {CONTENT\_TYPE\_FORMED})  
  
@RequestMapping(value = "/create",method={RequestMethod.*POST*},consumes = {*CONTENT\_TYPE\_FORMED*})  
@ResponseBody  
  
  
public CommonReturnType createItem(@RequestParam(name="title")String title,  
 @RequestParam(name="description")String description,  
 @RequestParam(name="price")BigDecimal price,  
 @RequestParam(name="stock")Integer stock,  
 @RequestParam(name="imgUrl")String imgUrl) throws BusinessException {  
  
 //封装service请求用来创建商品  
 ItemModel itemModel=new ItemModel();  
 itemModel.setTitle(title);  
 itemModel.setDescription(description);  
 itemModel.setPrice(price);  
 itemModel.setStock(stock);  
 itemModel.setImgUrl(imgUrl);  
  
 //ItemModel itemModelForReturn = (ItemModel) ItemService.createItem(itemModel);  
  
 ItemModel itemModelForReturn = ItemService.createItem(itemModel);  
 ItemVO itemVO=convertVOFromModel(itemModelForReturn);  
  
 return CommonReturnType.*create*(itemVO);

原因是，没有将 ItemService 注入 ItemController.java中  
@Autowired  
private ItemService itemService;

**商品表与库存表的关联逻辑错误（已解决）：**

商品详情Item表格：id(主键)，title, price， description， sales，img\_url

商品库存表item\_stock表格： id(主键), stock, item\_id

@Override  
@Transactional  
public ItemModel createItem(ItemModel itemModel) throws BusinessException {  
  
 //校验入参  
 ValidationResult result=validator.validate(itemModel);  
 if(result.isHasErrors()){  
  
 throw new BusinessException(EmBusinessError.*PARAMETER\_VALIDATION\_ERROR*,result.getErrMsg());  
  
 }  
  
 //转化itemmodel->dataobject  
 ItemDO itemDO=this.convertItemDOFromItemModel(itemModel);  
  
 //写入数据库  
 int id=itemDOMapper.insertSelective(itemDO);  
 System.*out*.println("Id:"+id); //此处id为成功插入数据库数据条数，成功插入为1  
 itemModel.setId(itemDO.getId()); //用来绑的库存表的itemid，逻辑错误：itemDO中id默认为null，而不是主键自增  
 ItemStockDO itemStockDO=this.convertItemStockDOFromItemModel(itemModel);  
 itemStockDOMapper.insertSelective(itemStockDO);  
  
 //返回创建完成的对象  
 return this.getItemById(itemModel.getId());  
  
  
}

Mybatis-generate.xml 配置：

<configuration>  
 <!--允许移动生产文件-->  
 <verbose>true</verbose>  
 <!--允许文件自动覆盖，实际开发中千万不要设置为 true-->  
 <overwrite>false</overwrite>  
 <!--generator 配置文件路径-->  
<!-- <configurationFile>  
 D:\JAVA\_CODE\SpringCode\dianshangmiaosha2\src\main\resources\mybatis-generator.xml  
 </configurationFile>-->  
 </configuration>

IDEA 报错:找不到包或者找不到符号

删除不必要的import导入

<https://blog.csdn.net/u013985664/article/details/79636638?utm_medium=distribute.pc_aggpage_search_result.none-task-blog-2~all~first_rank_v2~rank_v25-3-79636638.nonecase>

# 项目重难点：

框架的分层设计的思想

Spring boot 容器的IOC控制反转和AOP面向切片编程

注解的原理和应用

根据业务设计数据库表格的关联

# 后续需要深入学习的技术点：

Spring boot 控制反转的原理 (设计模式)

Spring boot 分层思想的理解和灵活应用

数据库表格的设计

IDEA 的断点调试和排错能力

Java基础

Mybatis、Hiberate、Redis、elasticsearch,kafka,rocketMq等

事务的特性

http生命周期

tcp/ip

抓包

# 提问：

Xml文件中Insert 和insertseletive的区别

insertSelective对应的sql语句加入了NULL校验，即只会插入数据不为null的字段值。  
insert则会插入所有字段，会插入null。

https://www.jianshu.com/p/c86fd116a28a

SQL查询数据库中空对象返回什么

Null

事务的原子性 适用于单个数据表格吗？

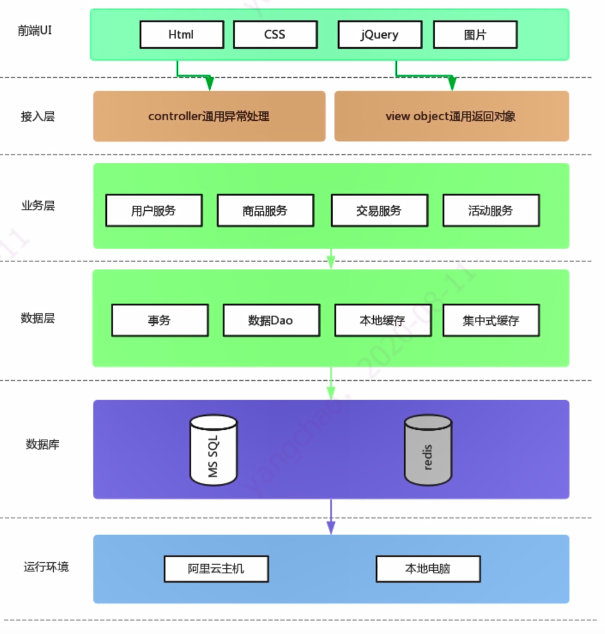
<https://blog.csdn.net/fanxing1964/article/details/82252774>

事务回滚：Spring boot中如果代码出错对数据库的操作会回滚吗？

Sql语句出错一般会回滚，其他代码出错一般不会回滚，最好用try catch语言抛出异常。

课程总结：

框架优化



后续待优化的方向：

