苍穹外卖day8

1. 导入地址簿模块功能代码

产品原型：

业务功能：

- 查询地址列表
- 新增地址
- 修改地址
- 删除地址
- 设置默认地址
- 查询默认地址

接口设计：

新增地址

查询当前登录用户的所有地址信息

查询默认地址

根据id修改地址

根据id刷除地址

根据id查询地址

设置默认地址

接口设计：新增地址

基本信息

Path: /user/addressBook

Method: POST

接口描述:

请求参数

Headers

参数名称	参数值	是否必须	示例	备注
Content-Type	application/json	是		

Body

名称	类型	是否必须	默认值	备注
cityCode	string	非必须		
cityName	string	非必须		
consignee	string	非必须		
detail	string	必须		详细地址
districtCode	string	非必须		
districtName	string	非必须		
id	integer	非必须		
isDefault	integer	非必须		
label	string	非必须		
phone	string	必须		手机号
provinceCode	string	非必须		
provinceName	string	非必须		
sex	string	必须		
userId	integer	非必须		

返回数据

名称	类型	是否必须	默认值	备注	其他信息
code	integer	必须			format: int32
data	object	非必须			
msg	string	非必须			

接口设计：查询登录用户所有地址

基本信息

Path： /user/addressBook/list

Method： GET

接口描述：

请求参数

返回数据

名称	类型	是否必须	默认值	备注	其他信息
code	integer	必须			format: int32
data	object	非必须			
├─ id	number	必须			
├─ userId	number	必须			
├─ consignee	string	必须			
├─ phone	string	必须			
├─ sex	string	必须			
├─ provinceCode	string	必须			
├─ provinceName	string	必须			
├─ cityCode	string	必须			
├─ cityName	string	必须			
├─ districtCode	string	必须			
├─ districtName	string	必须			
├─ detail	string	必须			
├─ label	string	必须			
├─ isDefault	number	必须			
msg	string	非必须			

接口设计：查询默认地址

基本信息

Path： /user/addressBook/default

Method： GET

接口描述：

请求参数

返回数据

名称	类型	是否必须	默认值	备注	其他信息
code	integer	必须			format: int32
data	object	非必须			
├─ cityCode	string	非必须			
├─ cityName	string	非必须			
├─ consignee	string	非必须			
├─ detail	string	非必须			
├─ districtCode	string	非必须			
├─ districtName	string	非必须			
├─ id	integer	非必须			format: int64
├─ isDefault	integer	非必须			format: int32
├─ label	string	非必须			
├─ phone	string	非必须			
├─ provinceCode	string	非必须			
├─ provinceName	string	非必须			
├─ sex	string	非必须			
├─ userId	integer	非必须			format: int64
msg	string	非必须			

接口设计：修改地址

基本信息

Path： /user/addressBook

Method： PUT

接口描述：

请求参数

Headers

参数名称	参数值	是否必须	示例	备注
Content-Type	application/json	是		

Body

名称	类型	是否必须	默认值	备注	其他信息
cityCode	string	非必须			
cityName	string	非必须			
consignee	string	非必须			
detail	string	必须		详细地址	
districtCode	string	非必须			
districtName	string	非必须			
id	integer	必须		主键值	format: int64
isDefault	integer	非必须			format: int32
label	string	非必须			
phone	string	必须		手机号	
provinceCode	string	非必须			
provinceName	string	非必须			
sex	string	必须			
userId	integer	非必须			format: int64

返回数据

名称	类型	是否必须	默认值	备注	其他信息
code	integer	必须			format: int32
data	object	非必须			
msg	string	非必须			

接口设计：根据id删除地址

基本信息

Path： /user/addressBook

Method： DELETE

接口描述：

请求参数

Query

参数名称	是否必须	示例	备注
id	是	101	地址id

返回数据

名称	类型	是否必须	默认值	备注	其他信息
code	integer	必须			format: int32
data	object	非必须			
msg	string	非必须			

接口设计：根据id查询地址

基本信息

Path： /user/addressBook/{id}

Method： GET

接口描述：

请求参数

路径参数

参数名称	示例	备注
id	101	地址id

返回数据

名称	类型	是否必须	默认值	备注	其他信息
code	number	必须			
data	object	必须			
├─ id	number	非必须			
├─ phone	string	非必须			
├─ consignee	string	非必须			
├─ userId	number	非必须			
├─ cityCode	string	非必须			
├─ provinceName	string	非必须			
├─ provinceCode	string	非必须			
├─ sex	string	非必须			
├─ districtName	string	非必须			
├─ districtCode	string	非必须			
├─ cityName	string	非必须			
├─ isDefault	number	非必须			
├─ label	string	非必须			
├─ detail	string	非必须			
msg	string	非必须			

接口设计：设置默认地址

基本信息

Path： /user/addressBook/default

Method： PUT

接口描述：

请求参数

Headers

参数名称	参数值	是否必须	示例	备注
Content-Type	application/json	是		

Body

名称	类型	是否必须	默认值	备注	其他信息
id	integer	必须		地址id	format: int64

返回数据

名称	类型	是否必须	默认值	备注	其他信息
code	integer	必须			format: int32
data	object	非必须			
msg	string	非必须			

数据库主要使用address_book表。

将C:\baidunetdiskdownload\资料\day08\地址簿模块功能代码 导入到

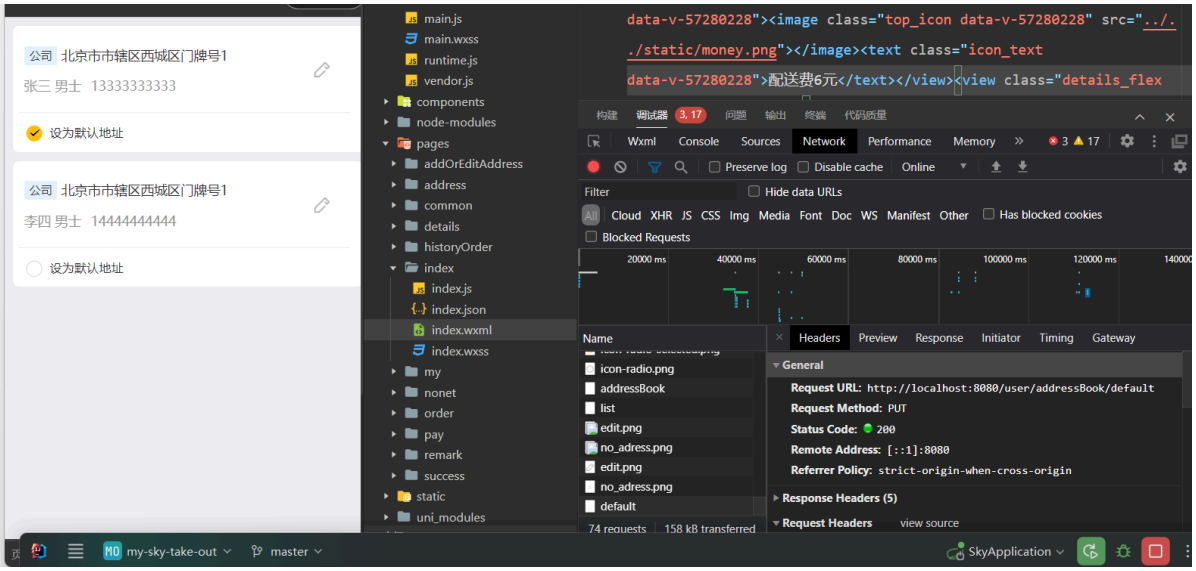
sky-server/src/main/java/com/sky/controller/user/AddressBookController.java

sky-server/src/main/java/com/sky/service/AddressBookService.java

sky-server/src/main/java/com/sky/service/impl/AddressBookServiceImpl.java

sky-server/src/main/java/com/sky/mapper/AddressBookMapper.java

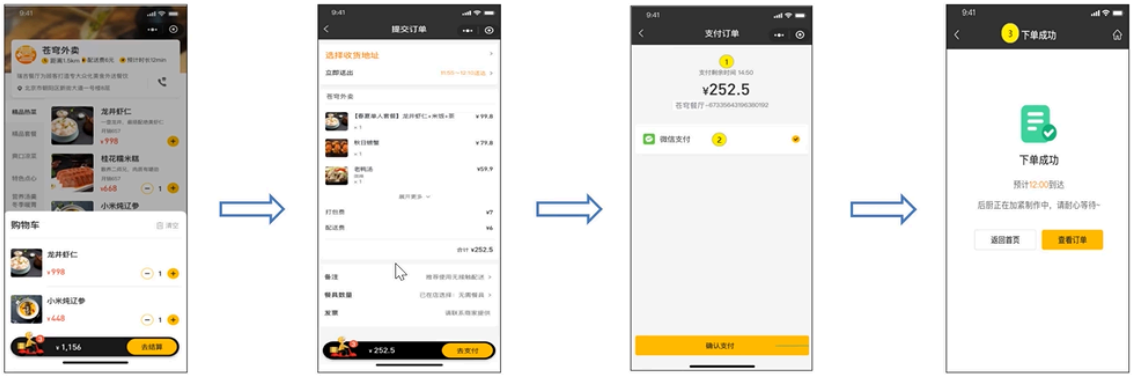
sky-server/src/main/resources/mapper/AddressBookMapper.xml



2. 用户下单

1. 需求分析和接口设计

用户点餐业务流程：



购物车页面　　订单提交页面　　订单支付页面　　下单成功页面

接口设计：

基本信息

Path: /user/order/submit

Method: POST

接口描述:

请求参数

Headers

参数名称	参数值	是否必须	示例	备注
Content-Type	application/json	是		

Body

名称	类型	是否必须	默认值	备注
addressBookId	integer	必须		地址簿id
amount	number	必须		总金额
deliveryStatus	integer	必须		配送状态： 1立即送出 0选择具体时间
estimatedDeliveryTime	string	必须		预计送达时间
packAmount	integer	必须		打包费
payMethod	integer	必须		付款方式
remark	string	必须		备注
tablewareNumber	integer	必须		餐具数量
tablewareStatus	integer	必须		餐具数量状态 1按餐量提供 0选择具体数量

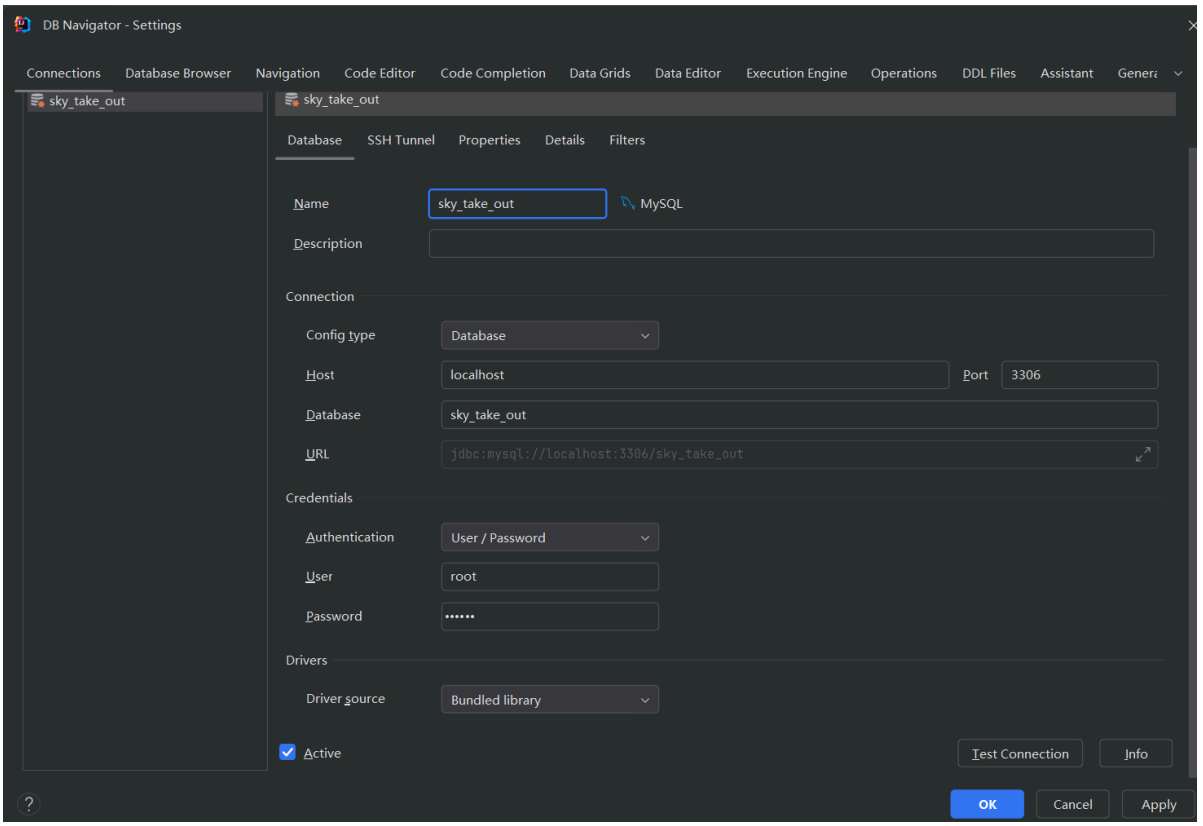
返回数据

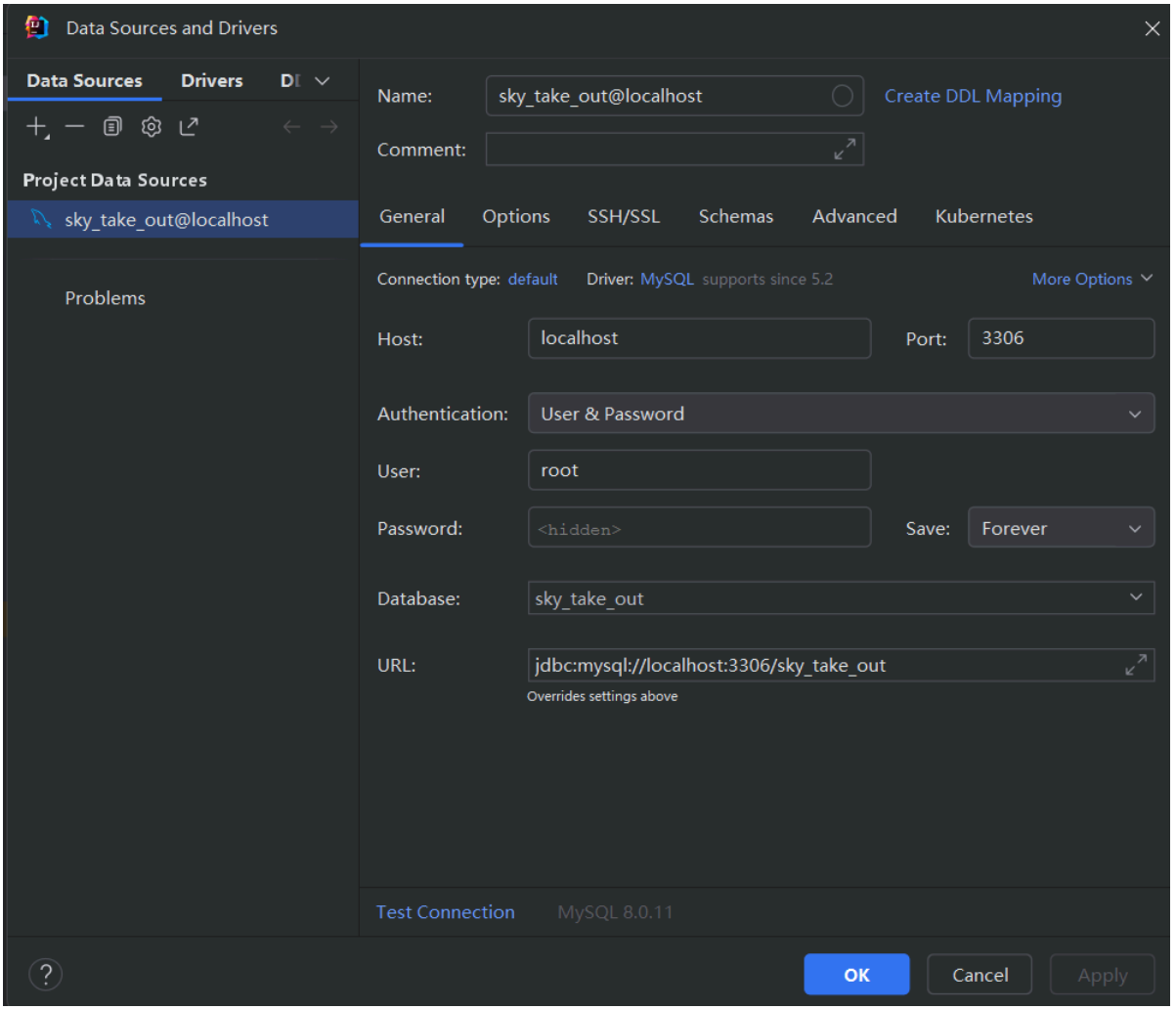
名称	类型	是否必须	默认值	备注
code	integer	必须		
data	object	必须		
├─id	integer	必须		订单id
├─orderAmount	number	必须		订单金额
├─orderNumber	string	必须		订单号
├─orderTime	string	必须		下单时间
msg	string	非必须		

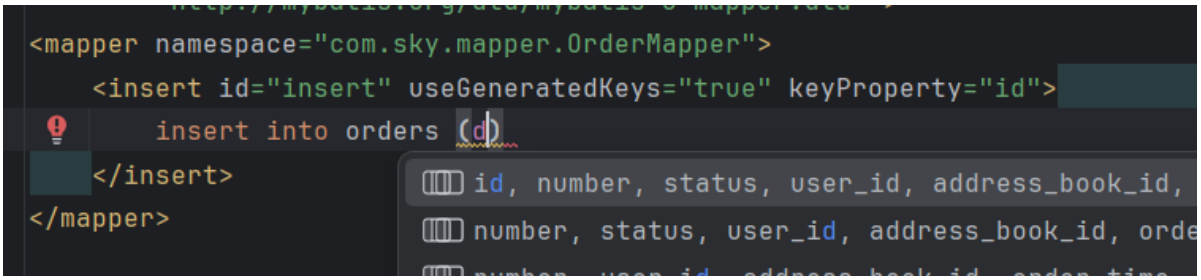
数据表主要看订单表orders，订单明细表order_detail

订单表和订单明细表关系：一对多

2. 解决idea数据表字段代码提示问题


DB Navigator - Settings
Connections
Database Browser
Navigation
Code Editor
Code Completion
Data Grids
Data Editor
Execution Engine
Operations
DDL Files
Assistant
sky_take_out
Database
SSH Tunnel
Properties
Details
Filters
Name
sky_take_out
MySQL
Description
Connection
Config type
Database
Host
localhost
Port
3306
Database
sky_take_out
URL
jdbc:mysql://localhost:3306/sky_take_out
Credentials
Authentication
User / Password
User
root
Password
Drivers
Driver source
Bundled library
Active
Test Connection
Info
OK
Cancel
Apply


Data Sources and Drivers
Data Sources
Drivers
Project Data Sources
sky_take_out@localhost
Problems
Name:
sky_take_out@localhost
Create DDL Mapping
Comment:
General
Options
SSH/SSL
Schemas
Advanced
Kubernetes
Connection type: default
Driver: MySQL supports since 5.2
More Options
Host:
localhost
Port:
3306
Authentication:
User & Password
User:
root
Password:
<hidden>
Save:
Forever
Database:
sky_take_out
URL:
jdbc:mysql://localhost:3306/sky_take_out
Overrides settings above
Test Connection
MySQL 8.0.11
OK
Cancel
Apply



3. 代码编写

sky-server/src/main/java/com/sky/controller/user/OrderController.java

```java
package com.sky.controller.user;

import com.sky.dto.OrdersSubmitDTO;
import com.sky.result.Result;
import com.sky.service.OrderService;
import com.sky.vo.OrderSubmitVO;
import io.swagger.annotations.Api;
import io.swagger.annotations.ApiOperation;
import lombok.extern.slf4j.Slf4j;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController("userOrderController")
@RequestMapping("/user/order")
@Api(tags="用户端订单相关接口")
@Slf4j
public class OrderController {

    @Autowired
    private OrderService orderService;

    /**
     * 用户下单
     * @param orderSubmitDTO
     * @return
     */
    @PostMapping("/submit")
    @ApiOperation("用户下单")
    public Result<OrderSubmitVO> submitOrder(@RequestBody OrdersSubmitDTO orderSubmitDTO) {
        log.info("用户下单，参数为:{}", orderSubmitDTO);
        OrderSubmitVO orderSubmitVO=orderService.submitOrder(orderSubmitDTO);
        return Result.success(orderSubmitVO);
    }
}
```

```
38
```

sky-server/src/main/java/com/sky/service/OrderService.java

```java
package com.sky.service;


import com.sky.dto.OrdersSubmitDTO;
import com.sky.vo.OrderSubmitVO;

public interface OrderService {


    /**
     * 用户下单
     * @param orderSubmitDTO
     * @return
     */
    OrderSubmitVO submitOrder(OrdersSubmitDTO orderSubmitDTO);
}

```

sky-server/src/main/java/com/sky/service/impl/OrderServiceImpl.java

```java
package com.sky.service.impl;

import com.sky.constant.MessageConstant;
import com.sky.context.BaseContext;
import com.sky.dto.OrdersSubmitDTO;
import com.sky.entity.AddressBook;
import com.sky.entity.OrderDetail;
import com.sky.entity.Orders;
import com.sky.entity.ShoppingCart;
import com.sky.exception.AddressBookBusinessException;
import com.sky.mapper.AddressBookMapper;
import com.sky.mapper.OrderDetailMapper;
import com.sky.mapper.OrderMapper;
import com.sky.mapper.ShoppingCartMapper;
import com.sky.service.OrderService;
import com.sky.vo.OrderSubmitVO;
import lombok.extern.slf4j.Slf4j;
import org.springframework.beans.BeanUtils;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;


import java.time.LocalDateTime;
import java.util.ArrayList;
import java.util.List;

```

```java
/**
 * 套餐业务实现
 */
@Service
@Slf4j
public class OrderServiceImpl implements OrderService {


    @Autowired
    private OrderMapper orderMapper;
    @Autowired
    private OrderDetailMapper orderDetailMapper;
    @Autowired
    private AddressBookMapper addressBookMapper;
    @Autowired
    private ShoppingCartMapper shoppingCartMapper;

    /**
     * 用户下单
     * @param orderSubmitDTO
     * @return
     */
    @Override
    @Transactional //采用事务，防止某一个sql语句出错导致数据不一致
    public OrderSubmitVO submitOrder(OrdersSubmitDTO orderSubmitDTO) {
        //1.处理各种业务异常（地址簿为空，购物车数据为空）
        AddressBook addressBook=addressBookMapper.getById(orderSubmitDTO.getAddressBookId());
        if(addressBook==null){
            //抛出业务异常
            throw new AddressBookBusinessException(MessageConstant.ADDRESS_BOOK_IS_NULL);
        }

        //查询当前用户的购物车数据
        Long userId= BaseContext.getCurrentId();
        ShoppingCart shoppingCart=new ShoppingCart();
        shoppingCart.setUserId(userId);
        List<ShoppingCart> shoppingCartList=shoppingCartMapper.list(shoppingCart);
        if(shoppingCartList==null || shoppingCartList.size()==0){
            //抛出业务异常
            throw new AddressBookBusinessException(MessageConstant.SHOPPING_CART_IS_NULL);
        }

        // 2.向订单表插入一条数据
        Orders orders=new Orders();
        BeanUtils.copyProperties(orderSubmitDTO,orders);
        orders.setOrderTime(LocalDateTime.now());
        orders.setPayStatus(Orders.UN_PAID);
        orders.setStatus(Orders.PENDING_PAYMENT);
        orders.setNumber(String.valueOf(System.currentTimeMillis()));//订单号
        orders.setPhone(addressBook.getPhone());
        orders.setConsignee(addressBook.getConsignee());//收货人
```

```
        orders.setUserId(userId);

        orderMapper.insert(orders);

        List<OrderDetail> orderDetailList=new ArrayList<>();
        // 3.向订单明细表插入n条数据
        for(ShoppingCart cart:shoppingCartList){
            OrderDetail orderDetail=new OrderDetail();
            BeanUtils.copyProperties(cart,orderDetail);
            orderDetail.setOrderId(orders.getId());
            orderDetailList.add(orderDetail);
        }
        orderDetailMapper.insertBatch(orderDetailList);
        // 4.清空当前用户的购物车数据
        shoppingCartMapper.deleteByUserId(userId);
        // 5.封装VO返回结果
        OrderSubmitVO orderSubmitVO=OrderSubmitVO.builder()
                .id(orders.getId())
                .orderTime(orders.getOrderTime())
                .orderNumber(orders.getNumber())
                .orderAmount(orders.getAmount())
                .build();

        return orderSubmitVO;
    }
}


```

sky-server/src/main/java/com/sky/mapper/OrderMapper.java

```java
package com.sky.mapper;

import com.sky.entity.Orders;
import org.apache.ibatis.annotations.Mapper;

@Mapper
public interface OrderMapper {

    /**
     * 订单表插入一条数据
     *
     * @param orders
     */
    void insert(Orders orders);
}

```

sky-server/src/main/resources/mapper/OrderMapper.xml

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
        "http://mybatis.org/dtd/mybatis-3-mapper.dtd" >
<mapper namespace="com.sky.mapper.OrderMapper">
    <insert id="insert" parameterType="Orders" useGeneratedKeys="true"
keyProperty="id">
        insert into orders
        (`number`, status, user_id, address_book_id, order_time,
checkout_time, pay_method, pay_status, amount, remark, phone, address,
user_name, consignee, cancel_reason, rejection_reason, cancel_time,
estimated_delivery_time, delivery_status, delivery_time, pack_amount,
tableware_number, tableware_status)
        values
            (#{number}, #{status}, #{userId}, #{addressBookId}, #
{orderTime}, #{checkoutTime}, #{payMethod}, #{payStatus}, #{amount}, #
{remark}, #{phone}, #{address}, #{userName}, #{consignee}, #{cancelReason},
#{rejectionReason}, #{cancelTime}, #{estimatedDeliveryTime}, #
{deliveryStatus}, #{deliveryTime}, #{packAmount}, #{tablewareNumber}, #
{tablewareStatus})
    </insert>
</mapper>

```

sky-server/src/main/java/com/sky/mapper/OrderDetailMapper.java

```java
package com.sky.mapper;

import com.sky.entity.OrderDetail;
import org.apache.ibatis.annotations.Mapper;

import java.util.List;

@Mapper
public interface OrderDetailMapper {
    /**
     * 批量插入订单明细数据
     * @param orderDetailList
     */
    void insertBatch(List<OrderDetail> orderDetailList);
}

```

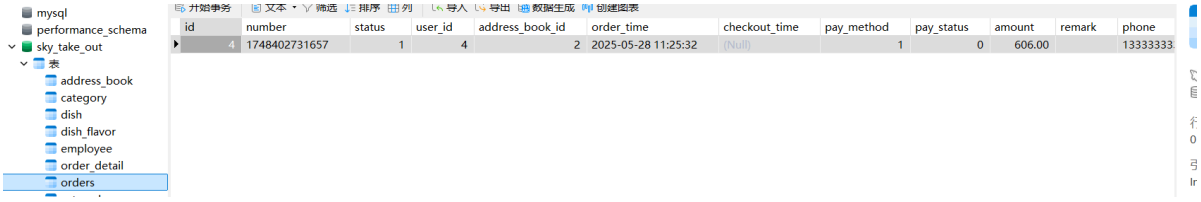
sky-server/src/main/resources/mapper/OrderDetailMapper.xml

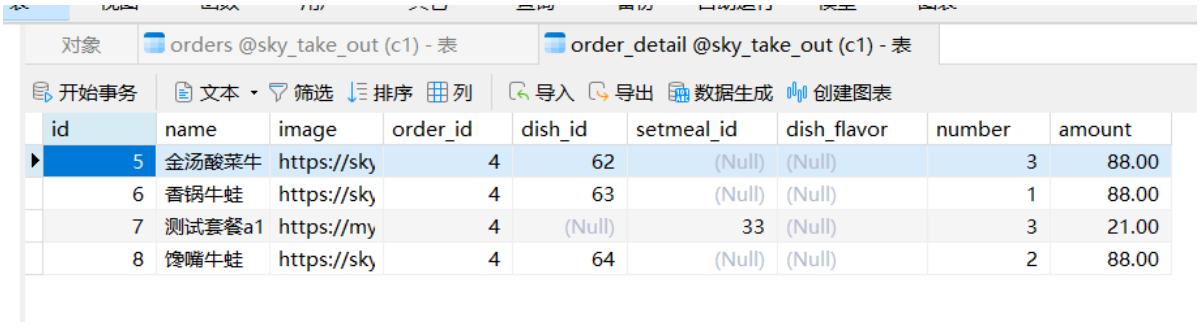
```xml
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
        "http://mybatis.org/dtd/mybatis-3-mapper.dtd" >
<mapper namespace="com.sky.mapper.OrderDetailMapper">
    <insert id="insertBatch">
        insert into order_detail

```

```
        (name, image, order_id, dish_id, setmeal_id, dish_flavor, `number`, amount)
        values
        <foreach collection="orderDetailList" item="od" separator=",">
            (#{od.name}, #{od.image}, #{od.orderId}, #{od.dishId}, #{od.setmealId}, #{od.dishFlavor}, #{od.number}, #{od.amount})
        </foreach>
    </insert>
</mapper>
```

微信开发者工具点击购物车，点击去支付，可看到数据库内容更新了





3. 订单支付

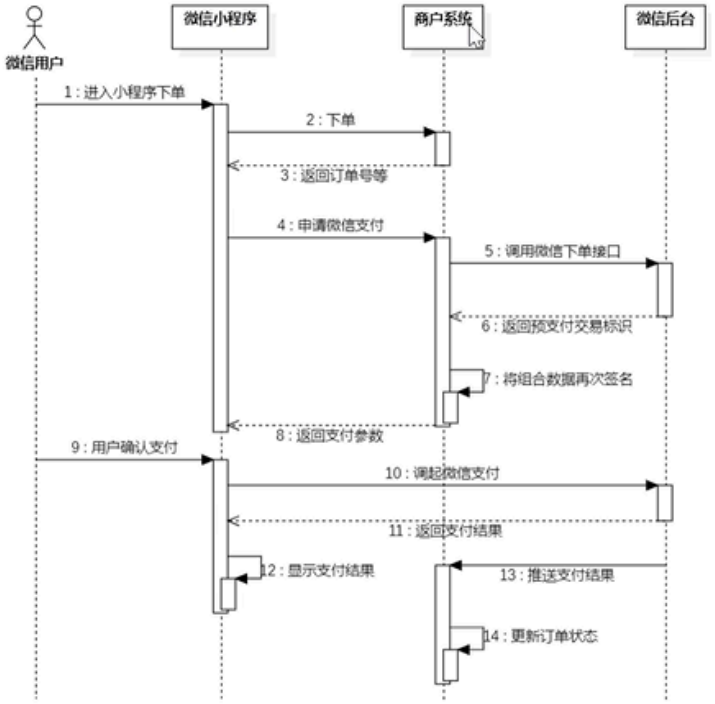
1. 微信支付介绍

微信支付文档：

https://pay.weixin.qq.com/static/product/product_index.shtml

微信支付接入流程：

微信小程序支付时序图：



请求示例

示例

```
wx.requestPayment
(
  {
    "timeStamp": "1414561699",
    "nonceStr": "5K8264ILTKCH16CQ2502SI8ZNMTM67VS",
    "package": "prepay_id=wx20141027200939552265 7a690389285100",
    "signType": "RSA",
    "paySign":
"oR9d8PuhnIc+YZ8cBHFCwfgpaK9gd7vaRvkYD7rthRAZ\/X+QBhcCYL21N7cHCTUxbQ+EAt6Uy+lwSN22f5YZvI
45MLko8Pfso0jm46v5hqcVwrk6uddkGuT+Cdvu4WBqDzaDjnNa5UK3GfE1Wfl2gHxIIY5lLdUgWFts17D4WuolLL
kiFZV+JSHMvH7eaLdT9N5GBovBwu5yYKUR7skR8Fu+LozcSqQixnlEZUfyE55feLOQTUYzLmR9pNtPbPsu6WVhbN
HMS3Ss2+AehHvz+n64GDmXxbX++IOBvm2olHu3PsOUGRwhudhVf7UcGcunXt8cqNjKNqZLhLw4jq\/xDg==",
    "success":function(res){},
    "fail":function(res){},
    "complete":function(res){}
  }
)
```

JSAPI下单：商户系统调用该接口在微信支付服务后台生成预支付交易单

适用对象： 直连商户

请求URL： https://api.mch.weixin.qq.com/v3/pay/transactions/jsapi

请求方式： POST

```
{
  "mchid": "1900006XXX",
  "out_trade_no": "1217752501201407033233368318",
  "appid": "wxdace645e0bc2cXXX",
  "description": "Image形象店-深圳腾大-QQ公仔",
  "notify_url": "https://www.weixin.qq.com/wxpay/pay.php",
  "amount": {
    "total": 1,
    "currency": "CNY"
  },
  "payer": {
    "openid": "o4GgauInH_RCEdvrrNGrntXDuXXX"
  }
}
```

返回参数

参数名	变量	类型[长度限制]	必填	描述
预支付交易会话标识	prepay_id	string[1,64]	是	预支付交易会话标识。用于后续接口调用中使用，该值有效期为2小时 示例值：wx20141027200939552265 7a690389285100

微信小程序调起支付：通过JSAPI下单接口获取到发起支付的必要参数prepay_id，然后使用微信支付提供的小程序方法调起小程序支付

接口名称：wx.requestPayment，详见小程序API文档

Object参数说明：

参数名	变量	类型[长度限制]	必填	描述
时间戳	timeStamp	string[1,32]	是	当前的时间，其他详见时间戳规则。 示例值：1414561699
随机字符串	nonceStr	string[1,32]	是	随机字符串，不长于32位。 示例值：5K8264ILTKCH16CQ2502SI8ZNMTM67VS
订单详情扩展字符串	package	string[1,128]	是	小程序下单接口返回的prepay_id参数值，提交格式如：prepay_id=*** 示例值：prepay_id=wx20141027200939552265 7a690389285100
签名方式	signType	string[1,32]	是	签名类型，默认为RSA，仅支持RSA。 示例值：RSA
签名	paySign	string[1,512]	是	签名，使用字段appid、timeStamp、nonceStr、package计算得出的签名值 示例值：oR9d8Puhnlc+YZ8cBHFCwfgpaK9gd7vaRvkYD7rthRAZ\/X+QBhcCYL21N7cHCTUxbQ+EAt6Uy+lwSN22f5YZvI45MLko8Pfso0jm46v5hqcVwrk6uddkGuT+Cdvu4WBqDzaDjnNa5UK3GfE1Wfl2gHxIIY5lLdUgWFts17D4WuolLkiFZV+JS

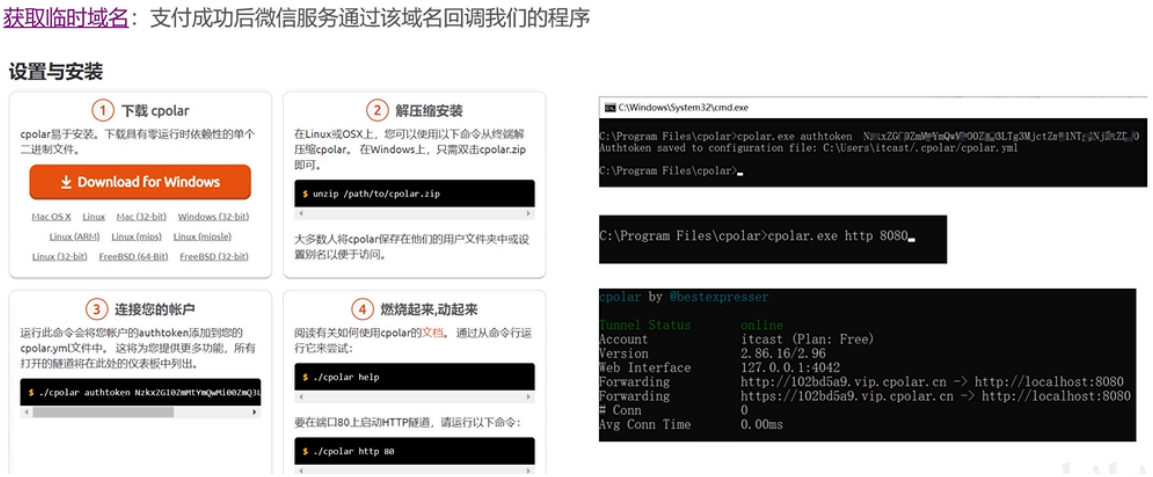
请求示例

示例

```
wx.requestPayment
(
  {
    "timeStamp": "1414561699",
    "nonceStr": "5K8264ILTKCH16CQ2502SI8ZNMTM67VS",
    "package": "prepay_id=wx20141027200939552265 7a690389285100",
    "signType": "RSA",
    "paySign":
"oR9d8PuhnIc+YZ8cBHFCwfgpaK9gd7vaRvkYD7rthRAZ\/X+QBhcCYL21N7cHCTUxbQ+EAt6Uy+lwSN22f5YZvI
45MLko8Pfso0jm46v5hqcVwrk6uddkGuT+Cdvu4WBqDzaDjnNa5UK3GfE1Wfl2gHxIIY5lLdUgWFts17D4WuolLL
kiFZV+JSHMvH7eaLdT9N5GBovBwu5yYKUR7skR8Fu+LozcSqQixnlEZUfyE55feLOQTUYzLmR9pNtPbPsu6WVhbN
HMS3Ss2+AehHvz+n64GDmXxbX++IOBvm2olHu3PsOUGRwhudhVf7UcGcunXt8cqNjKNqZLhLw4jq\/xDg==",
    "success":function(res){},
    "fail":function(res){},
    "complete":function(res){}
  }
)
```

2. 使用内网穿透工具

小程序还是往本地发数据，只是后端可以公网访问。



下载cpolar，访问https://dashboard.cpolar.com/get-started注册账号xian

先运行skyapplication，再运行cpolar.exe

```
C:\Program Files\cpolar> cpolar.exe authtoken
OTE5ZDkyNTktMGI0NS00OTFkLWIwODYtYmZmYzEwMjA3ZTVk
C:\Program Files\cpolar> cpolar.exe http 8080
```

访问http://1c168cb4.r10.cpolar.top/doc.html

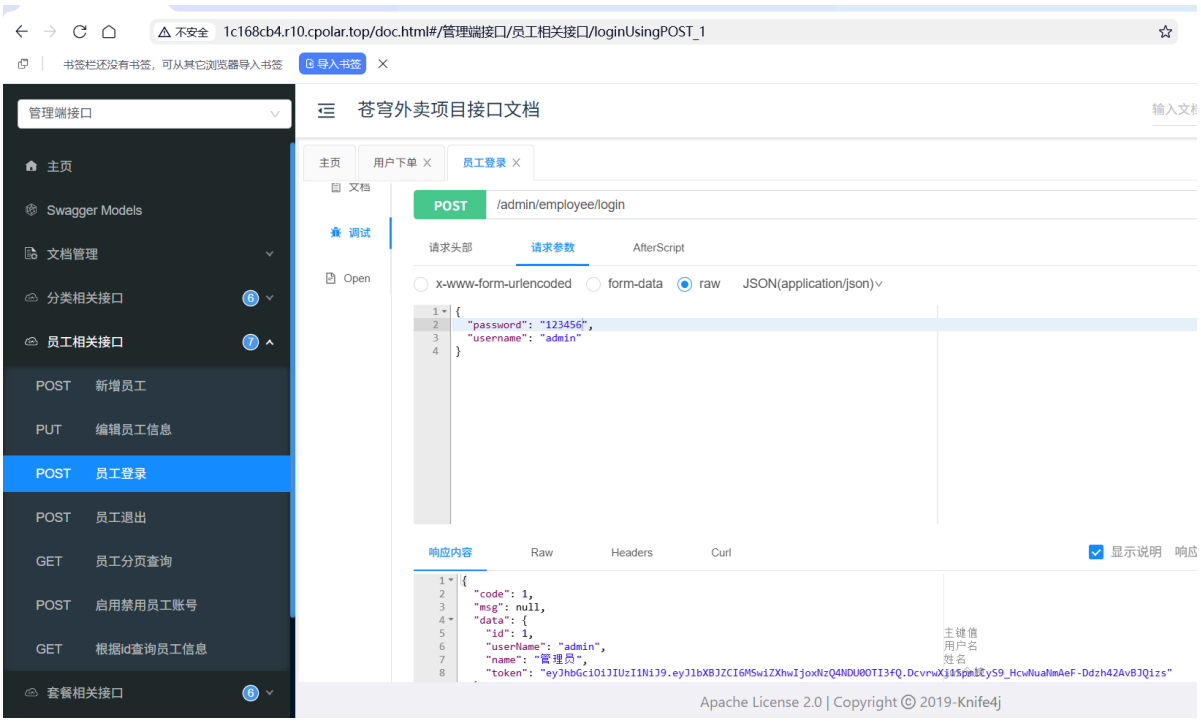
```
cpolar by @bestexpresser

Tunnel Status       online
Account             eyja6 (Plan: Free)
Version             2.86.16/3.18
Web Interface       127.0.0.1:4040
Forwarding          http://1c168cb4.r10.cpolar.top -> http://localhost:8080
Forwarding          https://1c168cb4.r10.cpolar.top -> http://localhost:8080
# Conn              0
Avg Conn Time       0.00ms


HTTP Requests
-------------

GET /v2/api-docs                200
GET /swagger-resources          200
GET /webjars/js/chunk-2d0af44 200
GET /webjars/js/chunk-214218f 200
GET /webjars/js/chunk-0fd6771 200
GET /webjars/css/chunk-vendor 200
GET /webjars/css/app.ff80f646 200
GET /doc.html                   200
GET /favicon.ico                404
```

看到可以在外网访问我们的内网服务



3. 代码导入

application.yml

```yaml
sky:
  wechat: #微信登录所需配置项
    mchid: ${sky.wechat.mchid}
    mchSerialNo: ${sky.wechat.mchid}
    privateKeyFilePath: ${sky.wechat.privateKeyFilePath}
    apiV3Key: ${sky.wechat.apiV3Key}
    weChatPayCertFilePath: ${sky.wechat.weChatPayCertFilePath}
    notifyUrl: ${sky.wechat.notifyUrl}
    refundNotifyUrl: ${sky.wechat.refundNotifyUrl}
```

application-dev.yml

```yaml
sky:
  wechat:
    mchid: 1561414331 #我们前几项随便填，因为拿不到商户号
    mchSerialNo: 4B3B3DC35414AD50B1B755BAF8DE9CC7CF407606
    privateKeyFilePath: C:\software\apiclient_key.pem
    apiV3Key: CZBK51236435wxpay435434323FFDuv3
    weChatPayCertFilePath:
C:\software\wechatpay_166D96F876F45C7D07CE98952A96EC980368ACFC.pem
    notifyUrl: http://1c168cb4.r10.cpolar.top/notify/paySuccess #前半部分是
cpolar返回的公网ip地址，
    refundNotifyUrl: http://1c168cb4.r10.cpolar.top/notify/refundSuccess
```

OrderController.java

```java
    /**
     * 订单支付
     *
     * @param ordersPaymentDTO
     * @return
     */
    @PutMapping("/payment")
    @ApiOperation("订单支付")
    public Result<OrderPaymentVO> payment(@RequestBody OrdersPaymentDTO
ordersPaymentDTO) throws Exception {
        log.info("订单支付：{}", ordersPaymentDTO);
        OrderPaymentVO orderPaymentVO =
orderService.payment(ordersPaymentDTO);
        log.info("生成预支付交易单：{}", orderPaymentVO);
        return Result.success(orderPaymentVO);
    }
```

OrderService.java

```java
    /**
     * 订单支付
     * @param ordersPaymentDTO
     * @return
     */
    OrderPaymentVO payment(OrdersPaymentDTO ordersPaymentDTO) throws Exception;

    /**
     * 支付成功，修改订单状态
     * @param outTradeNo
     */
    void paySuccess(String outTradeNo);
```

OrderServiceImpl.java

```java
    @Autowired
    private UserMapper userMapper;
    @Autowired
    private WeChatPayUtil weChatPayUtil;


    /**
     * 订单支付
     *
     * @param ordersPaymentDTO
     * @return
     */
    public OrderPaymentVO payment(OrdersPaymentDTO ordersPaymentDTO) throws Exception {
        // 当前登录用户id
        Long userId = BaseContext.getCurrentId();
        User user = userMapper.getById(userId);

        //调用微信支付接口，生成预支付交易单
        JSONObject jsonObject = weChatPayUtil.pay(
                ordersPaymentDTO.getOrderNumber(), //商户订单号
                new BigDecimal(0.01), //支付金额，单位 元
                "苍穹外卖订单", //商品描述
                user.getOpenid() //微信用户的openid
        );

        if (jsonObject.getString("code") != null && jsonObject.getString("code").equals("ORDERPAID")) {
            throw new OrderBusinessException("该订单已支付");
        }

        OrderPaymentVO vo = jsonObject.toJavaObject(OrderPaymentVO.class);
        vo.setPackageStr(jsonObject.getString("package"));

        return vo;
    }

```

```
    /**
     * 支付成功，修改订单状态
     *
     * @param outTradeNo
     */
    public void paySuccess(String outTradeNo) {

        // 根据订单号查询订单
        Orders ordersDB = orderMapper.getByNumber(outTradeNo);

        // 根据订单id更新订单的状态、支付方式、支付状态、结账时间
        Orders orders = Orders.builder()
                .id(ordersDB.getId())
                .status(Orders.TO_BE_CONFIRMED)
                .payStatus(Orders.PAID)
                .checkoutTime(LocalDateTime.now())
                .build();

        orderMapper.update(orders);
    }
```

OrderMapper.java

```
    /**
     * 根据订单号查询订单
     * @param orderNumber
     */
    @Select("select * from orders where number = #{orderNumber}")
    Orders getByNumber(String orderNumber);

    /**
     * 修改订单信息
     * @param orders
     */
    void update(Orders orders);
```

OrderMapper.xml

```
    <update id="update" parameterType="com.sky.entity.Orders">
        update orders
        <set>
            <if test="cancelReason != null and cancelReason!='' ">
                cancel_reason=#{cancelReason},
            </if>
            <if test="rejectionReason != null and rejectionReason!='' ">
                rejection_reason=#{rejectionReason},
            </if>
            <if test="cancelTime != null">
                cancel_time=#{cancelTime},
            </if>
            <if test="payStatus != null">
```

```
                pay_status=#{payStatus},
            </if>
            <if test="payMethod != null">
                pay_method=#{payMethod},
            </if>
            <if test="checkoutTime != null">
                checkout_time=#{checkoutTime},
            </if>
            <if test="status != null">
                status = #{status},
            </if>
            <if test="deliveryTime != null">
                delivery_time = #{deliveryTime}
            </if>
        </set>
        where id = #{id}
    </update>
```

sky-server/src/main/java/com/sky/controller/notify/PayNotifyController.java

```
package com.sky.controller.notify;

import com.alibaba.druid.support.json.JSONUtils;
import com.alibaba.fastjson.JSON;
import com.alibaba.fastjson.JSONObject;
import com.sky.properties.WeChatProperties;
import com.sky.service.OrderService;
import com.wechat.pay.contrib.apache.httpclient.util.AesUtil;
import lombok.extern.slf4j.Slf4j;
import org.apache.http.entity.ContentType;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.BufferedReader;
import java.nio.charset.StandardCharsets;
import java.util.HashMap;

/**
 * 支付回调相关接口
 */
@RestController
@RequestMapping("/notify")
@Slf4j
public class PayNotifyController {
    @Autowired
    private OrderService orderService;
    @Autowired
    private WeChatProperties weChatProperties;

    /**
     * 支付成功回调
```

```
     *
     * @param request
     */
    @RequestMapping("/paySuccess")
    public void paySuccessNotify(HttpServletRequest request, HttpServletResponse response) throws Exception {
        //读取数据
        String body = readData(request);
        log.info("支付成功回调：{}", body);

        //数据解密
        String plainText = decryptData(body);
        log.info("解密后的文本：{}", plainText);

        JSONObject jsonObject = JSON.parseObject(plainText);
        String outTradeNo = jsonObject.getString("out_trade_no");//商户平台订单号
        String transactionId = jsonObject.getString("transaction_id");//微信支付交易号

        log.info("商户平台订单号：{}", outTradeNo);
        log.info("微信支付交易号：{}", transactionId);

        //业务处理，修改订单状态、来单提醒
        orderService.paySuccess(outTradeNo);

        //给微信响应
        responseToWeixin(response);
    }

    /**
     * 读取数据
     *
     * @param request
     * @return
     * @throws Exception
     */
    private String readData(HttpServletRequest request) throws Exception {
        BufferedReader reader = request.getReader();
        StringBuilder result = new StringBuilder();
        String line = null;
        while ((line = reader.readLine()) != null) {
            if (result.length() > 0) {
                result.append("\n");
            }
            result.append(line);
        }
        return result.toString();
    }

    /**
     * 数据解密
     *
     * @param body
     * @return
     * @throws Exception
```

```
     */
    private String decryptData(String body) throws Exception {
        JSONObject resultObject = JSON.parseObject(body);
        JSONObject resource = resultObject.getJSONObject("resource");
        String ciphertext = resource.getString("ciphertext");
        String nonce = resource.getString("nonce");
        String associatedData = resource.getString("associated_data");

        AesUtil aesUtil = new
AesUtil(weChatProperties.getApiV3Key().getBytes(StandardCharsets.UTF_8));
        //密文解密
        String plainText =
aesUtil.decryptToString(associatedData.getBytes(StandardCharsets.UTF_8),
                nonce.getBytes(StandardCharsets.UTF_8),
                ciphertext);

        return plainText;
    }

    /**
     * 给微信响应
     * @param response
     */
    private void responseToWeixin(HttpServletResponse response) throws
Exception{
        response.setStatus(200);
        HashMap<Object, Object> map = new HashMap<>();
        map.put("code", "SUCCESS");
        map.put("message", "SUCCESS");
        response.setHeader("Content-type",
ContentType.APPLICATION_JSON.toString());

 response.getOutputStream().write(JSONUtils.toJSONString(map).getBytes(Stan
dardCharsets.UTF_8));
        response.flushBuffer();
    }
}

```

UserMapper.java

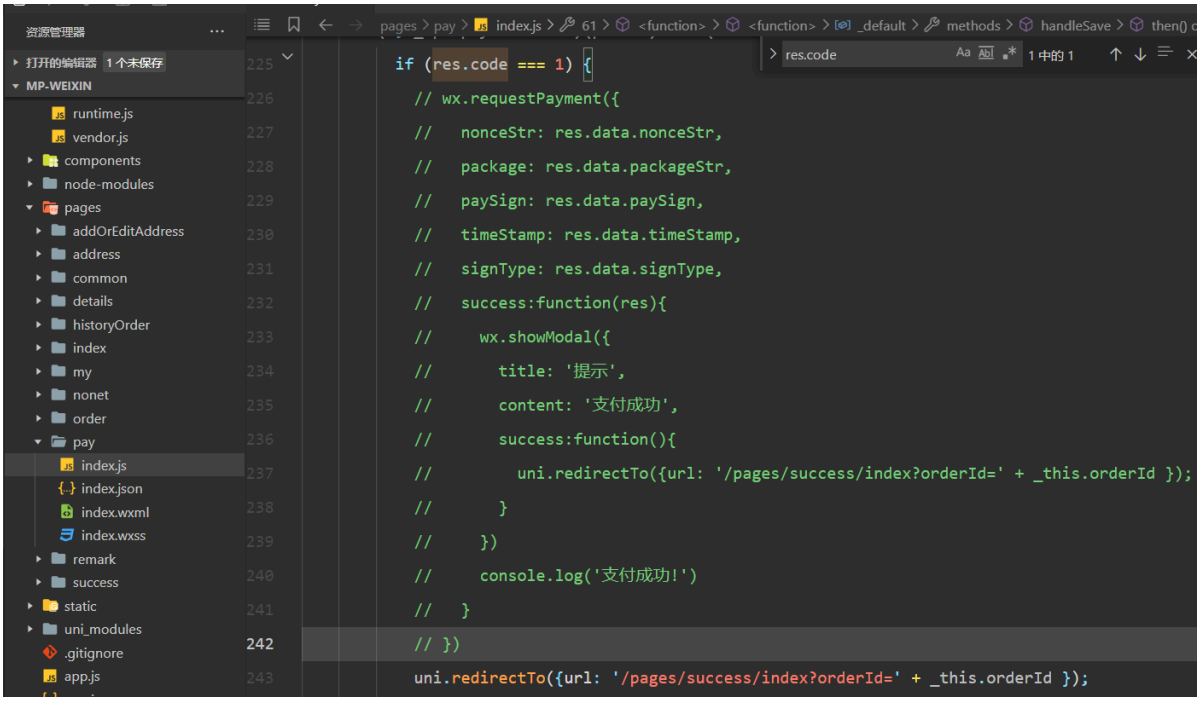
```
    /**
     * 根据用户id查询用户
     *
     * @param userId
     * @return
     */
    @Select("select * from user where id = #{userId}")
    User getById(Long userId);
```

4. 绕过订单支付

魔改参考

[苍穹外卖笔记 吾沐西风]

小程序直接重定向，绕过微信支付



OrderServiceImpl.java注意修改位置

```
    private Orders orders;//添加位置1，用作全局参数

        // 2.向订单表插入一条数据
        Orders orders=new Orders();
        BeanUtils.copyProperties(orderSubmitDTO,orders);
        orders.setOrderTime(LocalDateTime.now());
        orders.setPayStatus(Orders.UN_PAID);
        orders.setStatus(Orders.PENDING_PAYMENT);
        orders.setNumber(String.valueOf(System.currentTimeMillis()));//订单号
        orders.setPhone(addressBook.getPhone());
        orders.setConsignee(addressBook.getConsignee());//收货人
        orders.setUserId(userId);
        this.orders=orders;//添加位置2
        orderMapper.insert(orders);

        List<OrderDetail> orderDetailList=new ArrayList<>();

//注释以下代码
//调用微信支付接口，生成预支付交易单
//        JSONObject jsonObject = weChatPayUtil.pay(
//                ordersPaymentDTO.getOrderNumber(), //商户订单号
//                new BigDecimal(0.01), //支付金额，单位 元
//                "苍穹外卖订单", //商品描述
//                user.getOpenid() //微信用户的openid
```

```
//        );
//
//        if (jsonObject.getString("code") != null &&
jsonObject.getString("code").equals("ORDERPAID")) {
//            throw new OrderBusinessException("该订单已支付");
//        }

//添加以下代码
        JSONObject jsonObject = new JSONObject();
        jsonObject.put("code","ORDERPAID");
        OrderPaymentVO vo = jsonObject.toJavaObject(OrderPaymentVO.class);
        vo.setPackageStr(jsonObject.getString("package"));
        Integer OrderPaidStatus = Orders.PAID;//支付状态，已支付
        Integer OrderStatus = Orders.TO_BE_CONFIRMED;  //订单状态，待接单
        LocalDateTime check_out_time = LocalDateTime.now();//更新支付时间
        orderMapper.updateStatus(OrderStatus, OrderPaidStatus,
check_out_time, this.orders.getId());
        return vo;
```

OrderMapper.java

```
    @Update("update orders set status = #{orderStatus},pay_status = #
{orderPaidStatus} ,checkout_time = #{check_out_time} where id = #{id}")
    void updateStatus(Integer orderStatus, Integer orderPaidStatus,
LocalDateTime check_out_time, Long id);
```