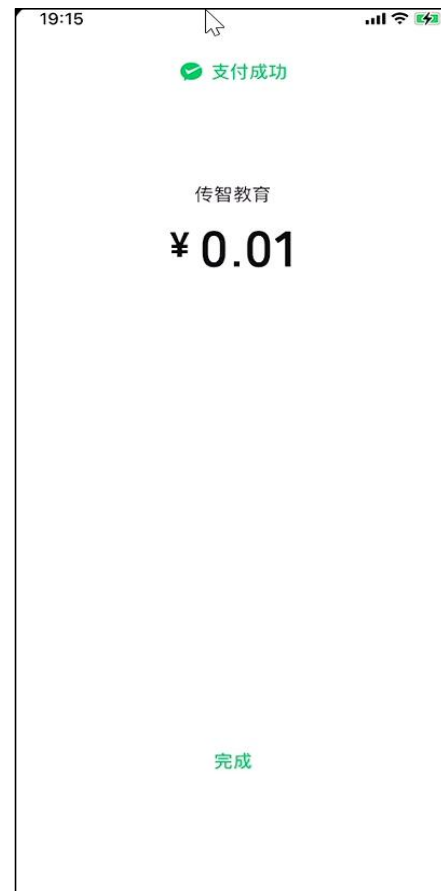
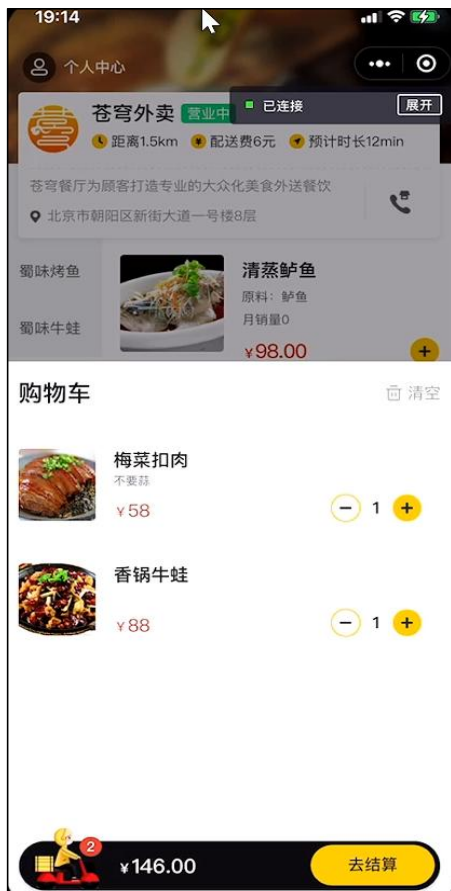


用户下单、订单支付



黑马程序员
www.itheima.com

传智教育旗下
高端IT教育品牌





目录

Contents

- ◆ 导入地址簿功能代码
- ◆ 用户下单
- ◆ 订单支付

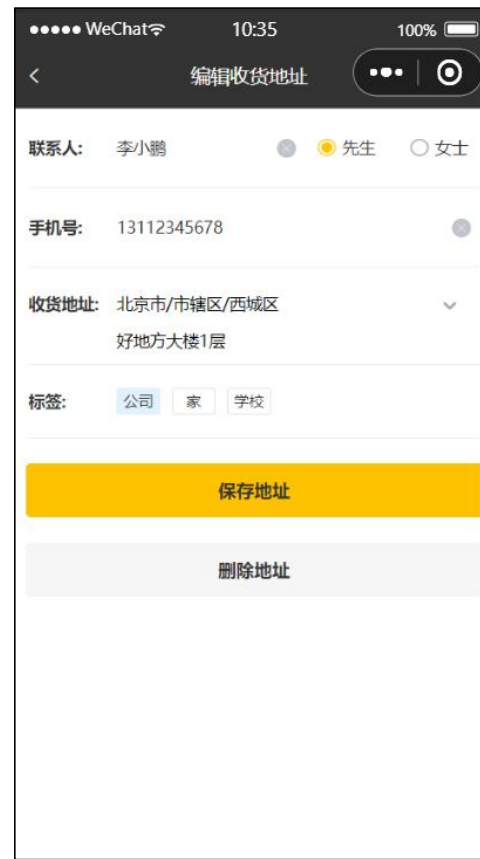


导入地址簿功能代码

- 需求分析和设计
- 代码导入
- 功能测试

需求分析和设计

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

需求分析和设计

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删除地址

根据id查询地址

设置默认地址

需求分析和设计

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			

需求分析和设计

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			

需求分析和设计

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			

需求分析和设计

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			

需求分析和设计

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			

需求分析和设计

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			

需求分析和设计

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			

需求分析和设计

数据库设计（address_book表）：

字段名	数据类型	说明	备注
id	bigint	主键	自增
user_id	bigint	用户id	逻辑外键
consignee	varchar(50)	收货人	
sex	varchar(2)	性别	
phone	varchar(11)	手机号	
province_code	varchar(12)	省份编码	
province_name	varchar(32)	省份名称	
city_code	varchar(12)	城市编码	
city_name	varchar(32)	城市名称	
district_code	varchar(12)	区县编码	
district_name	varchar(32)	区县名称	
detail	varchar(200)	详细地址信息	具体到门牌号
label	varchar(100)	标签	公司、家、学校
is_default	tinyint(1)	是否默认地址	1是 0否



导入地址簿功能代码






- 需求分析和设计
- 代码导入
- 功能测试

代码导入

导入课程资料中的地址簿模块功能代码：

苍穹外卖项目 > 授课资料 > day08-用户下单、订单支付 > 资料 > 地址簿模块功能代码

名称

-  AddressBookController.java
-  AddressBookMapper.java
-  AddressBookMapper.xml
-  AddressBookService.java
-  AddressBookServiceImpl.java



导入地址簿功能代码

- 需求分析和设计
- 代码导入
- 功能测试

功能测试

可以通过如下方式进行测试：

- 查看控制台sql和数据库中的数据变化
- Swagger接口文档测试
- 前后端联调



目录

Contents

- ◆ 导入地址簿功能代码
- ◆ 用户下单
- ◆ 订单支付



用户下单


- 需求分析和设计
- 代码开发
- 功能测试

需求分析和设计

用户下单业务说明：

在电商系统中，用户是通过下单的方式通知商家，用户已经购买了商品，需要商家进行备货和发货。

用户下单后会产生订单相关数据，订单数据需要能够体现如下信息：



订单总金额是多少？ ← 21.8元(已付款)

哪个用户下的单？ ← 曾先生

收货地址是哪？ → 金泰花苑宁南街13号
欣欣宾馆203

用户手机号是多少？ → 155-1541-1550

买的哪些商品？
每个商品数量是多少？ →

菜名	数量	小计
黑米粥	x1	3.00
鱼香肉丝盖饭	x1	14.00
餐盒费	x1	1.80
配送费	x1	3.00

需求分析和设计

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

需求分析和设计

接口设计（分析）：



请求方式：POST

请求路径：/user/order/submit

参数：

- 地址簿id
- 配送状态（立即送出、选择送出时间）
- 打包费
- 总金额
- 备注
- 餐具数量

需求分析和设计

接口设计（分析）：



返回数据：

- 下单时间
- 订单总金额
- 订单号
- 订单id

需求分析和设计

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		

需求分析和设计

数据库设计：

- 订单表 orders
- 订单明细表 order_detail

谁的订单?
送哪去?
打哪个电话联系?
多少钱?
什么时间下的单?
什么时间支付的?
订单的状态?
订单号是多少?

当前明细属于哪个订单?
具体点的是什么商品?
这个商品点了几份?

订单表和订单明细表的关系：一对多

需求分析和设计

数据库设计：订单表 orders

字段名	数据类型	说明	备注
id	bigint	主键	自增
number	varchar(50)	订单号	
status	int	订单状态	1待付款 2待接单 3已接单 4派送中 5已完成 6已取消
user_id	bigint	用户id	逻辑外键
address_book_id	bigint	地址id	逻辑外键
order_time	datetime	下单时间	
checkout_time	datetime	付款时间	
pay_method	int	支付方式	1微信支付 2支付宝支付
pay_status	tinyint	支付状态	0未支付 1已支付 2退款
amount	decimal(10,2)	订单金额	
remark	varchar(100)	备注信息	

phone	varchar(11)	手机号	冗余字段
address	varchar(255)	详细地址信息	冗余字段
consignee	varchar(32)	收货人	冗余字段
cancel_reason	varchar(255)	订单取消原因	
rejection_reason	varchar(255)	拒单原因	
cancel_time	datetime	订单取消时间	
estimated_delivery_time	datetime	预计送达时间	
delivery_status	tinyint	配送状态	1立即送出 0选择具体时间
delivery_time	datetime	送达时间	
pack_amount	int	打包费	
tableware_number	int	餐具数量	
tableware_status	tinyint	餐具数量状态	1按餐量提供 0选择具体数量

需求分析和设计

数据库设计：订单明细表 order_detail

字段名	数据类型	说明	备注
id	bigint	主键	自增
name	varchar(32)	商品名称	冗余字段
image	varchar(255)	商品图片路径	冗余字段
order_id	bigint	订单id	逻辑外键
dish_id	bigint	菜品id	逻辑外键
setmeal_id	bigint	套餐id	逻辑外键
dish_flavor	varchar(50)	菜品口味	
number	int	商品数量	
amount	decimal(10,2)	商品单价	



用户下单

- 需求分析和设计
- 代码开发
- 功能测试

代码开发

根据用户下单接口的参数设计DTO:

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



```
@Data
public class OrdersSubmitDTO implements Serializable {
    //地址簿id
    private Long addressBookId;
    //付款方式
    private int payMethod;
    //备注
    private String remark;
    //预计送达时间
    @JsonFormat(shape = JsonFormat.Shape.STRING, pattern = "yyyy-MM-dd HH:mm:ss")
    private LocalDateTime estimatedDeliveryTime;
    //配送状态 1立即送出 0选择具体时间
    private Integer deliveryStatus;
    //餐具数量
    private Integer tablewareNumber;
    //餐具数量状态 1按餐量提供 0选择具体数量
    private Integer tablewareStatus;
    //打包费
    private Integer packAmount;
    //总金额
    private BigDecimal amount;
}
```

代码开发

根据用户下单接口的返回结果设计VO:

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	必须			
└ id	integer	必须		订单id	format: int64
└ orderAmount	number	必须		订单金额	
└ orderNumber	string	必须		订单号	
└ orderTime	string	必须		下单时间	format: date-time
msg	string	非必须			



```
@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
public class OrderSubmitVO implements Serializable {
    //订单id
    private Long id;
    //订单号
    private String orderNumber;
    //订单金额
    private BigDecimal orderAmount;
    //下单时间
    private LocalDateTime orderTime;
}
```

代码开发

创建OrderController并提供用户下单方法：

```
@RestController("userOrderController")
@RequestMapping("/user/order")
@Slf4j
@Api(tags = "C端订单接口")
public class OrderController {

    @Autowired
    private OrderService orderService;

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


代码开发

创建OrderService接口，并声明用户下单方法：

```
public interface OrderService {  
  
    /**  
     * 用户下单  
     * @param ordersSubmitDTO  
     * @return  
     */  
    OrderSubmitVO submitOrder(OrdersSubmitDTO ordersSubmitDTO);  
}
```

代码开发

创建OrderServiceImpl实现OrderService接口 (1) :

```
/**
 * 订单
 */
@Service
@Slf4j
public class OrderServiceImpl implements OrderService {

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

代码开发

创建OrderServiceImpl实现OrderService接口（2）：

```
/**
 * 用户下单
 * @param ordersSubmitDTO
 * @return
 */
@Transactional
public OrderSubmitVO submitOrder(OrdersSubmitDTO ordersSubmitDTO) {
    //异常情况的处理（收货地址为空、购物车为空）
    AddressBook addressBook = addressBookMapper.getById(ordersSubmitDTO.getAddressBookId());
    if (addressBook == null) {
        throw new AddressBookBusinessException(MessageConstant.ADDRESS_BOOK_IS_NULL);
    }

    Long userId = BaseContext.getCurrentId();
    ShoppingCart shoppingCart = new ShoppingCart();
    shoppingCart.setUserId(userId);

    //查询当前用户的购物车数据
    List<ShoppingCart> shoppingCartList = shoppingCartMapper.list(shoppingCart);
    if (shoppingCartList == null || shoppingCartList.size() == 0) {
        throw new ShoppingCartBusinessException(MessageConstant.SHOPPING_CART_IS_NULL);
    }
}
```

代码开发

创建OrderServiceImpl实现OrderService接口 (3) :

```
//构造订单数据
Orders order = new Orders();
BeanUtils.copyProperties(ordersSubmitDTO,order);
order.setPhone(addressBook.getPhone());
order.setAddress(addressBook.getDetail());
order.setConsignee(addressBook.getConsignee());
order.setNumber(String.valueOf(System.currentTimeMillis()));
order.setUserId(userId);
order.setStatus(Orders.PENDING_PAYMENT);
order.setPayStatus(Orders.UN_PAID);
order.setOrderTime(LocalDate.now());

//向订单表插入1条数据
orderMapper.insert(order);
```

代码开发

创建OrderServiceImpl实现OrderService接口（4）：

```
//订单明细数据
List<OrderDetail> orderDetailList = new ArrayList<>();
for (ShoppingCart cart : shoppingCartList) {
    OrderDetail orderDetail = new OrderDetail();
    BeanUtils.copyProperties(cart, orderDetail);
    orderDetail.setOrderId(order.getId());
    orderDetailList.add(orderDetail);
}
//向明细表插入n条数据
orderDetailMapper.insertBatch(orderDetailList);

//清理购物车中的数据
shoppingCartMapper.deleteByUserId(userId);

//封装返回结果
OrderSubmitVO orderSubmitVO = OrderSubmitVO.builder()
    .id(order.getId())
    .orderNumber(order.getNumber())
    .orderAmount(order.getAmount())
    .orderTime(order.getOrderTime())
    .build();
return orderSubmitVO;
}
```

代码开发

创建OrderMapper接口和对应的xml映射文件：

```
@Mapper
public interface OrderMapper {
    /**
     * 插入订单数据
     * @param order
     */
    void insert(Orders order);
}
```

```
<?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, consignee, estimated_delivery_time, delivery_status,
        pack_amount, tableware_number, tableware_status)
        values
        (#{number}, #{status}, #{userId}, #{addressBookId}, #{orderTime}, #{checkoutTime}, #{payMethod}, #{payStatus},
        #{amount}, #{remark}, #{phone}, #{address}, #{consignee}, #{estimatedDeliveryTime}, #{deliveryStatus},
        #{packAmount}, #{tablewareNumber}, #{tablewareStatus})
    </insert>

</mapper>
```

代码开发

创建OrderDetailMapper接口和对应的xml映射文件：

```
@Mapper
public interface OrderDetailMapper {

    /**
     * 批量插入订单明细数据
     * @param orderDetails
     */
    void insertBatch(List<OrderDetail> orderDetails);
}
```

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

</mapper>
```



用户下单

- 需求分析和设计
- 代码开发
- 功能测试

功能测试

可以通过如下方式进行测试：

- 查看控制台sql
- 前后端联调
- 查看数据库中数据变化



目录

Contents

- ◆ 导入地址簿功能代码
- ◆ 用户下单
- ◆ 订单支付



订单支付

- 微信支付介绍
- 微信支付准备工作
- 代码导入
- 功能测试

微信支付介绍

微信支付产品：



付款码支付



JSAPI支付



小程序支付



Native支付



APP支付



刷脸支付

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

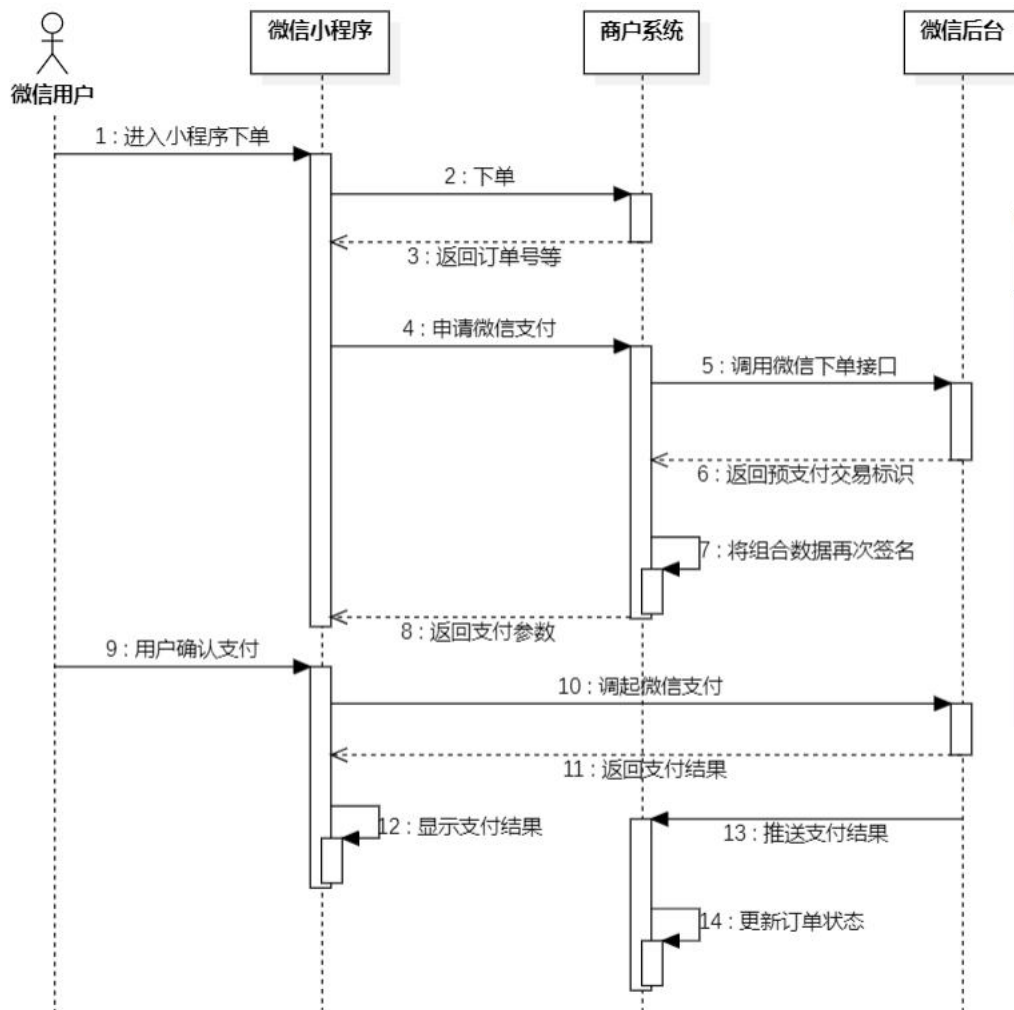
微信支付介绍

微信支付接入流程：



微信支付介绍

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=wx201410272009395522657a690389285100",
  "signType": "RSA",
  "paySign":
"oR9d8PuhnIc+YZ8cBHFcfwfgpaK9gd7vaRvkYD7rthRAZ\\X+QBhcCYL21N7cHCTUxbQ+EAt6Uy+lwSN22f5YZvI
45MLko8PFso0jm46v5hqcVvrk6uddkGuT+Cdvu4WBqDzaDjnNa5UK3GfE1WF12gHxIIY51LdUgWFTs17D4Wuo1LL
kiFZV+JSHMvH7eaLdT9N5GBovBwu5yYKUR7skR8Fu+LozcSqQixn1EZUfyE55feL0QTUYzLmR9pNtPbPsu6wVhbN
HMS3Ss2+AehHvz+n64GDmXxbX++IOBvm2o1Hu3PsOUGRwhudhVf7UcGcunXt8cqNjKNqZLhLw4jq\\x0g==",
  "success":function(res){},
  "fail":function(res){},
  "complete":function(res){}
})
```

微信支付介绍

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: wx201410272009395522657a6 90389285100

微信支付介绍

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=wx201410272009395522657a690389285100
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+Cdvu4WBqDzaDjnNa5UK3GfE1Wfl2gHxliY5ILdUgWfTs17D4WuoLLkiFZV+JS

请求示例

示例

```
wx.requestPayment
({
  "timeStamp": "1414561699",
  "nonceStr": "5K8264ILTKCH16CQ2502SI8ZNMTM67VS",
  "package": "prepay_id=wx201410272009395522657a690389285100",
  "signType": "RSA",
  "paySign":
  "oR9d8Puhnlc+YZ8cBHFCwfgpaK9gd7vaRvkYD7rthRAZ\X+QBhcCYL21N7cHCTUxbQ+EAt6Uy+lwSN22f5YZvI45MLko8Pfso0jm46v5hqcVwrk6uddkGuT+Cdvu4WBqDzaDjnNa5UK3GfE1Wfl2gHxliY5ILdUgWfTs17D4WuoLLkiFZV+JS",
  "success":function(res){},
  "fail":function(res){},
  "complete":function(res){}
})
```

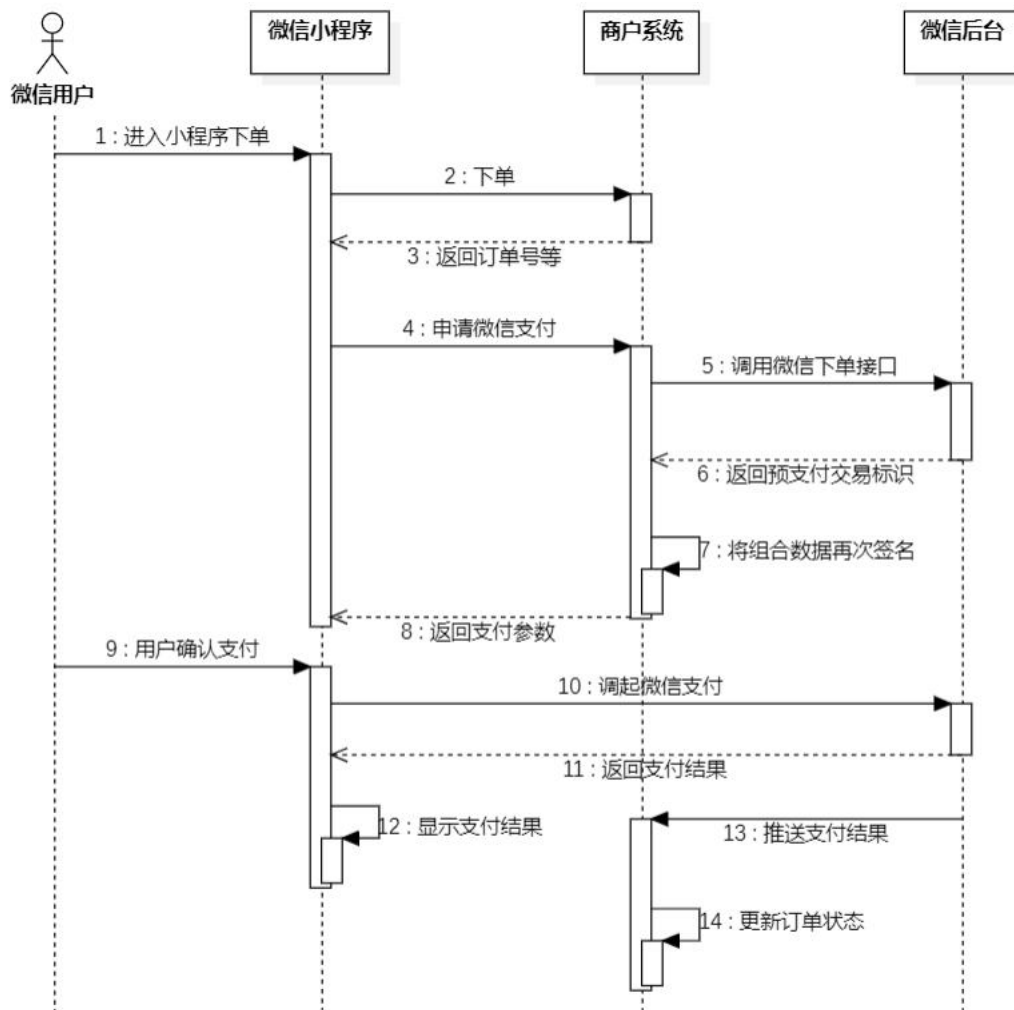



订单支付

- 微信支付介绍
- 微信支付准备工作
- 代码导入
- 功能测试

微信支付准备工作

微信小程序支付时序图：





调用过程如何保证数据安全?

微信后台如何调用到商户系统?

微信支付介绍

获取微信支付平台证书、商户私钥文件：

 apiclient_key.pem

 wechatpay_166D96F876F45C7D07CE98952A96EC980368ACFC.pem

微信支付准备工作

获取临时域名：支付成功后微信服务通过该域名回调我们的程序

设置与安装

① 下载 cpolar

cpolar易于安装。下载具有零运行时依赖性的单个二进制文件。

↓ Download for Windows

[Mac OS X](#) [Linux](#) [Mac \(32-bit\)](#) [Windows \(32-bit\)](#)

[Linux \(ARM\)](#) [Linux \(mips\)](#) [Linux \(mipsle\)](#)

[Linux \(32-bit\)](#) [FreeBSD \(64-Bit\)](#) [FreeBSD \(32-bit\)](#)

② 解压缩安装

在Linux或OSX上，您可以使用以下命令从终端解压缩cpolar。在Windows上，只需双击cpolar.zip即可。

```
$ unzip /path/to/cpolar.zip
```

大多数人将cpolar保存在他们的用户文件夹中或设置别名以便于访问。

③ 连接您的帐户

运行此命令会将您帐户的authtoken添加到您的cpolar.yml文件中。这将为提供更多功能，所有打开的隧道将在此处的仪表板中列出。

```
$ ./cpolar authtoken NzKxZGI0ZmMtYmQwMi00ZmQ3L
```

④ 燃烧起来,动起来

阅读有关如何使用cpolar的文档。通过从命令行运行它来尝试：

```
$ ./cpolar help
```

要在端口80上启动HTTP隧道，请运行以下命令：

```
$ ./cpolar http 80
```

```
C:\Windows\System32\cmd.exe
```

```
C:\Program Files\cpolar>cpolar.exe authtoken NzKxZGI0ZmMtYmQwMi00ZmQ3L3LTg3MjctZn1NTfNjZT0  
Auth token saved to configuration file: C:\Users\itcast/.cpolar/cpolar.yml
```

```
C:\Program Files\cpolar>
```

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

```
cpolar by @bestexpresser
```

```
Tunnel Status      online  
Account            itcast (Plan: Free)  
Version            2.86.16/2.96  
Web Interface      127.0.0.1:4042  
Forwarding         http://102bd5a9.vip.cpolar.cn -> http://localhost:8080  
Forwarding         https://102bd5a9.vip.cpolar.cn -> http://localhost:8080  
# Conn             0  
Avg Conn Time      0.00ms
```



订单支付

- 微信支付介绍
- 微信支付准备工作
- 代码导入
- 功能测试

代码导入

微信支付相关配置：

sky:

```
jwt: <6 keys>
alioss: <4 keys>
wechat:
  appid: ${sky.wechat.appid}
  secret: ${sky.wechat.secret}
  mchid : ${sky.wechat.mchid}
  mchSerialNo: ${sky.wechat.mchSerialNo}
  privateKeyFilePath: ${sky.wechat.privateKeyFilePath}
  apiV3Key: ${sky.wechat.apiV3Key}
  weChatPayCertFilePath: ${sky.wechat.weChatPayCertFilePath}
  notifyUrl: ${sky.wechat.notifyUrl}
  refundNotifyUrl: ${sky.wechat.refundNotifyUrl}
```

```
@Component
@ConfigurationProperties(prefix = "sky.wechat")
@Data
public class WeChatProperties {

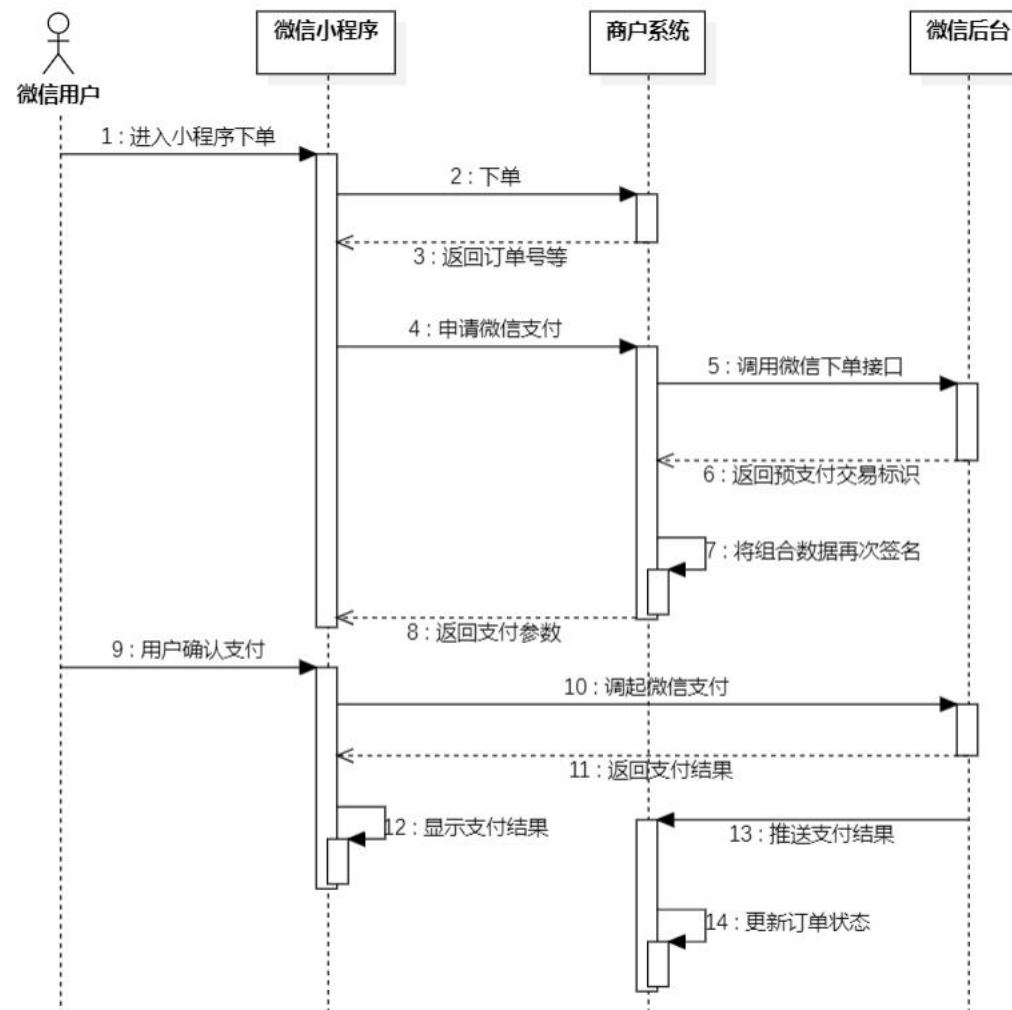
    private String appid; // 小程序的appid
    private String secret; // 小程序的密钥
    private String mchid; // 商户号
    private String mchSerialNo; // 商户API证书的证书序列号
    private String privateKeyFilePath; // 商户私钥文件
    private String apiV3Key; // 证书解密的密钥
    private String weChatPayCertFilePath; // 平台证书
    private String notifyUrl; // 支付成功的回调地址
    private String refundNotifyUrl; // 退款成功的回调地址

}
```

代码导入

导入微信支付功能代码：

- OrderController.java
- OrderMapper.java
- OrderMapper.xml
- OrderService.java
- OrderServiceImpl.java
- PayNotifyController.java



代码导入

微信支付相关配置：



传智教育旗下高端IT教育品牌