订单模块API说明文档

一、订单模块概述:

此为达达商城订单功能模块API说明文档,此文档对于常规业务逻辑的流程进行分析和针对订单模块的技术点进行分析。订单模块提供以下功能:

> 确认订单
>
> 生成订单并支付
>
> 支付订单
>
> 查看订单信息
>
> 用户确认收货

二、事件定义

1.确认订单

用户在购物车页面点击确认**去结算**按钮，前端跳转到确认订单页(orderconfirm.html)发送Ajax请求(get)到后端，后端从redis中获取**商品信息**和从mysql中获取**用户地址信息**响应给到订单确认页展示。

2.生成订单并支付

用户在确认订单页面点击**确认并付款**按钮，前端跳转到支付订单页(payment.html)发送Ajax请求(post)到后端,后端生成订单且将**订单编号、订单状态（默认为待支付)、订单金额**等信息响应给前端订单支付页展示.此次请求带回第三方支付的URL.用户在支付订单页点击**确认支付**按钮，前端获取事件使用**window.location**去访问第三方支付**pay_URL**,之后由支付宝提供支付业务。支付完成支付宝重定向到订单支付结果页面,并异步通知到商城后端。双重验证后返回支付结果给前端展示.

3.订单查询

用户在页面头部点击**我的订单图标,**前端跳转到**我的订单页**(Myorder.html)发送Ajax请求(get)到后端,后端查询订单数据相应给前端.前端展示依据订单的状态**全部订单、待付款、待发货、待收货、已完成**进行展示数据即**订单编号、订单金额、成交时间和交易状态。**

4.确认收货

用户在我的订单页面点击确认收货,前端获取点击事件向后端发送请求(get).后端将订单状态改为4，响应200

5.订单支付

用户在我的订单页面点击**去支付**按钮，前端跳转到支付订单页(payment.html)发送Ajax请求(post)到后端,后端生成订单且将**订单编号、订单状态（默认为待支付)、订单金额**等信息响应给前端订单支付页展示.此次请求带回第三方支付的URL.

三、API说明

1.确认订单API

用户在登录情况下才能够点击去结算并跳转到确认订单页.

1.事件触发:

点击去结算按钮

确认订单页展示数据

事件触发(异常):用户无收货地址

用户点击去结算按钮


请输入您要搜索的内容
帮助
qyjtedu
首页
私人定制
购物车
已选1件商品 合计(不含运费): 209元
全选
商品
单价(元)
数量
金额
操作
红袖添香
颜色：红色
尺寸：18寸
阿甲专享价
¥209.00
1
¥ 209
删除
全选
已选1件商品 合计(不含运费):209元
去结算

点击确定按钮


127.0.0.1 显示
地址为空,请添加地址
确定

`

确认订单页展示

2.请求:

- **URL**：http://127.0.0.1:8000/v1/orders/processing/
- **请求方式:**GET
- **请求参数:**

参数	类型	是否必须	说明
status	int	是	业务类型码
settlement	int	是	结算类型

- **请求示例:**

```
127.0.0.1:8000/v1/orders/processing/?status=0&settlement_type=1
```

3.响应:

- **返回值类型:**JSON
- **响应内容:**

字段	含义	类型	备注
code	状态码	int	默认正常为200，异常见状态码
data	具体数据	dict	与error二选一

- **响应格式:**

```
{"code":200,"data":data}
```

- **data参数信息:**

参数	类型	是否必须	说明
addresses	list	是	用户收货地址列表
sku_list	list	是	商品信息
total_count	int	是	商品总数量
total_amount	decimail	是	总价格
transit	int	是	运费
payment_amount	decimal	是	实付款

- **data数据示例:**

```
   [{"addresses":[],
      "sku_list":[],
      "total_count":2,
      "total_amount":100.00,
      "transit":10,
      "payment_amount":110
      }]
```

- **状态码参考:**

状态码	响应信息
200	正常
50100	用户未添加收货地址

4.代码实现:

购物车页中点击去结算按钮事件

```
$("#go-buy").click(function () {
    var username = window.localStorage.dashop_user;
    if (username == null){
        alert("请登录后结算")
        window.location.href = "login.html"
    }else{
        var arrOutput = [];
    $(".Each input:checkbox:checked").each(function (i, e) {
        $('.num1207').each(function (i2, e2) {
            if ($(e).val() == $(e2).data("id")) {
                var _obj = {}
                _obj.id = $(e2).data("id");
                _obj.num = $(e2).val()
                arrOutput.push(_obj)
            }
        })
    })
    window.location.href = "orderConfirm.html"
    }
})
```

确认订单页向后端请求数据

```
function push_ajax(parameter_url){
                $.ajax({
                    type: 'get',
                    // parameter_url参数字符串"?status=0&settlement_type=1"
                    // status 为业务类型代码
                    // settlement_type 为结算方式代码
                    url: baseUrl+'/v1/orders/processing/'+parameter_url,
                    datatype: 'json',
                    beforeSend:function(request){
                        request.setRequestHeader("authorization",token)
                    },
                    async: false,
                    success: function (response) {
                        // 渲染响应回来的数据
                        })
```

后端

```
    # 判断业务类型码为0即为确认订单业务
    if status == 0:
            # 1.获取收货地址列表
            addresses_list = self.get_address(request)
```

```
            # 2.组织商品数据
            settlement = int(request.GET.get('settlement_type'))
            # 3.判断结算方式
            # 购物车结算
            if settlement == 0:
            # 获取购买购物车商品列表
                cart_dict = self.get_cart_dict(request) #从redis中获取购物车中的商品
id
                skus = SKU.objects.filter(id__in=cart_dict.keys())
                # 获取商品列表，总数量,总价格,运费,实付款(总价格+运费)
                sku_list, total_count, total_amount, transit, payment_amount =
self.get_order_list(skus, cart_dict)
            # 直接购买结算
            elif settlement == 1:
                sku_id = request.GET.get('sku_id')
                count = request.GET.get('buy_num')
                skus = SKU.objects.filter(id=sku_id)
                cart_dict = None
                # 获取商品列表，总数量,总价格,运费,实付款(总价格+运费)
                sku_list, total_count, total_amount, transit, payment_amount =
self.get_order_list(skus,cart_dict,count)
            # 4.组织数据
            data = {
                'addresses': addresses_list,
                'sku_list': sku_list,
                'total_count': total_count,
                'total_amount': total_amount,
                'transit': transit,
                'payment_amount': payment_amount,
                'addresses':addresses_list
            }
            return http.JsonResponse({'code':200, 'data':data,
'base_url':PIC_URL})
```

2.生成订单API

1.事件触发:

点击确认并付款按钮

订单支付页展示

点击确认并支付

)

支付宝账户登录

输入支付密码

支付宝支付结果展示

dadashop支付结果展示

2.请求

- **URL：** http://127.0.0.1:8000/v1/orders/processing/
- **请求方式:**POST
- **请求参数:**

参数	类型	是否必须	说明
status	int	是	业务类型码
address_id	int	是	收获地址id

- **请求示例:**

```
http://127.0.0.1:8000/v1/orders/processing/?status=0&address_id=1
```

3.响应:

- **返回值类型:**JSON
- **响应内容:**

字段	含义	类型	备注
code	状态码	int	默认正常为200，异常见状态码
data	具体数据	dict	与error二选一
error	错误信息	char	与data二选一

- **响应格式:**

```
{"code":200,"data":data}
```

- **data参数信息:**

参数	类型	是否必须	说明

参数	类型	是否必须	说明
saller	str	是	商家名称
total_amount	decimail	是	总价格
order_id	str	是	订单号
pay_url	str	是	第三方支付路由

- **data数据示例:**

```
   [{'saller': '达达商城',
     'total_amount': total_amount + 10,
     'order_id': order_id,
     'pay_url': pay_url
      }]
```

- **error参数信息:**

参数	类型	是否必须	说明
error_msg	str	是	错误信息说明

- **error数据示例:**

```
[{
    "error_msg":""
}]
```

- **状态码参考:**

状态码	响应信息
200	正常
50104	操作太快,清扫后重试
50103	商品X库存不足
50102	收货地址无效

4.代码实现

确认订单页点击确认并付款

```
$("#go_pay").click(function () {
      var address_id = $('.base_select')[0].getAttribute("data-id")
      window.location=
     "payment.htmladdress_id="+address_id+"&settlement_type="+settlement_type

                        })
```

订单支付页发送ajax获取数据

```
$.ajax({
```

```
        type: 'post',
        url: baseUrl+'/v1/orders/processing/',
        beforeSend:function(request){
        request.setRequestHeader("authorization",token)
        },
        data:JSON.stringify({
            "status":0,
            "address_id":address_id,
            "settlement_type":settlement_type
        }),
        datatype: 'json',
        async: false,
        success: function (response) {
            //渲染响应回来的数据
        }
```

后端代码

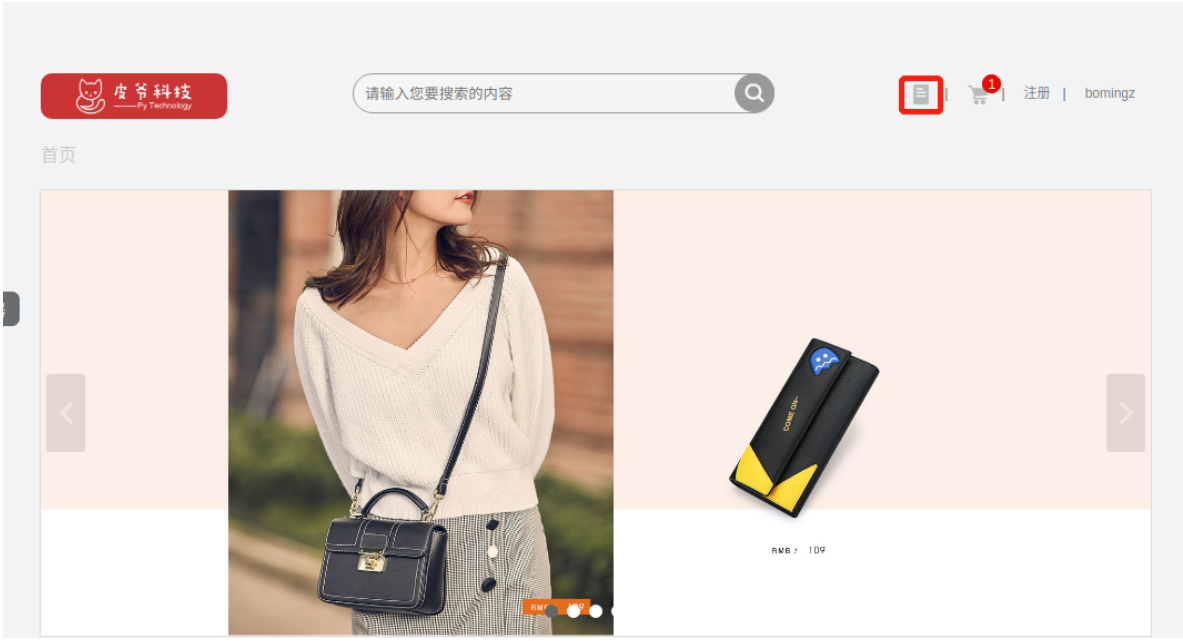
```
if status == 0:
    address_id = json.loads(request.body.decode()).get("address_id")
    settlement_type = json.loads(request.body.decode()).get("settlement_type")
    try:
        address = Address.objects.get(id=address_id, is_alive=True)
    except:
        return http.JsonResponse({'code': 50102, 'errmsg': '收货地址无效'})
    now = datetime.datetime.now()
    with transaction.atomic():  # 禁止自动提交
        # 开启事务
        sid = transaction.savepoint()
        # 1.创建订单基本对象
        order_id = '%s%02d' % (now.strftime('%Y%m%d%H%M%S'), user.id)
        total_count = 0
        total_amount = 0
        order = OrderInfo.objects.create(
            order_id=order_id,
            user_id=user.id,
            address_id=address_id,
            total_count=0,
            total_amount=0,
            freight=1,
            pay_method=1,
            status=1
        )


        # 2.数据库中修改商品的数量
        cart_dict = self.get_cart_dict(request) # 从redis中获取订单商品的信息.
        skus = SKU.objects.filter(id__in=cart_dict.keys())
        # 用以存储购物车中选中的商品,用作订单生成后删除redis中的商品.
        sku_ids = []
        for sku in skus:
            selected = int(cart_dict[sku.id]['state']['selected'])
            if selected == 0:
                continue
            cart_count = int(cart_dict[sku.id]['state']['count'])
            sku_ids.append(sku.id)
            # 判断库存，不足则提示,如果足够则继续执行
```

```
            if sku.stock < cart_count:
                # 回滚事务
                transaction.savepoint_rollback(sid)
                sku_ids.remove(sku.id)
                # 给出提示
                return http.JsonResponse({'code': 50103, 'errmsg': '商品[%d]库存不
足' % sku.id})
            # 使用乐观锁进行修改
            # 使用乐观锁修改商品数量
            stock_old = sku.stock
            stock_new = sku.stock - cart_count
            sales_new = sku.sales + cart_count
            result = SKU.objects.filter(pk=sku.id,
stock=stock_old).update(stock=stock_new, sales=sales_new)
            # result表示sql语句修改数据的个数
            if result == 0:
                # 库存发生变化，未成功购买
                transaction.savepoint_rollback(sid)
                sku_ids.remove(sku.id)
                return http.JsonResponse({'code': 50104, 'errmsg': '操作太快了,请
稍后重试'})
            # 创建订单商品对象
            order_sku = OrderGoods.objects.create(
                order_id=order_id,
                sku_id=sku.id,
                count=cart_count,
                price=sku.price
            )
            #  计算总金额、总数量
            total_count += cart_count
            total_amount += sku.price * cart_count
        # 修改订单对象的总金额、总数量
        order.total_count = total_count
        order.total_amount = total_amount + 10
        order.save()

        # 提交
        transaction.savepoint_commit(sid)

    # 3.从redis中删除所有已经加入了订单的商品
    redis_cli = get_redis_connection('cart')
    for sku_id in sku_ids:
        redis_cli.hdel('cart_%d' % user.id, sku_id)
    # 4.生成支付宝支付链接
    sku_goods = OrderGoods.objects.filter(order=order_id)
    order_string =
self.get_order_string(order.order_id,order.total_amount,sku_goods)
    # 构建让用户跳转的支付链接
    pay_url = "https://openapi.alipaydev.com/gateway.do?" + order_string
    # 5.组织数据
    data = {
        'saller': '达达商城',
        'total_amount': order.total_amount,
        'order_id': order_id,
        'pay_url': pay_url
    }
    # 响应
    return http.JsonResponse({'code': 200, 'data': data})
```

3.订单查询API

1.在登录状态下才可查看订单信息

任意页面header点击订单图标

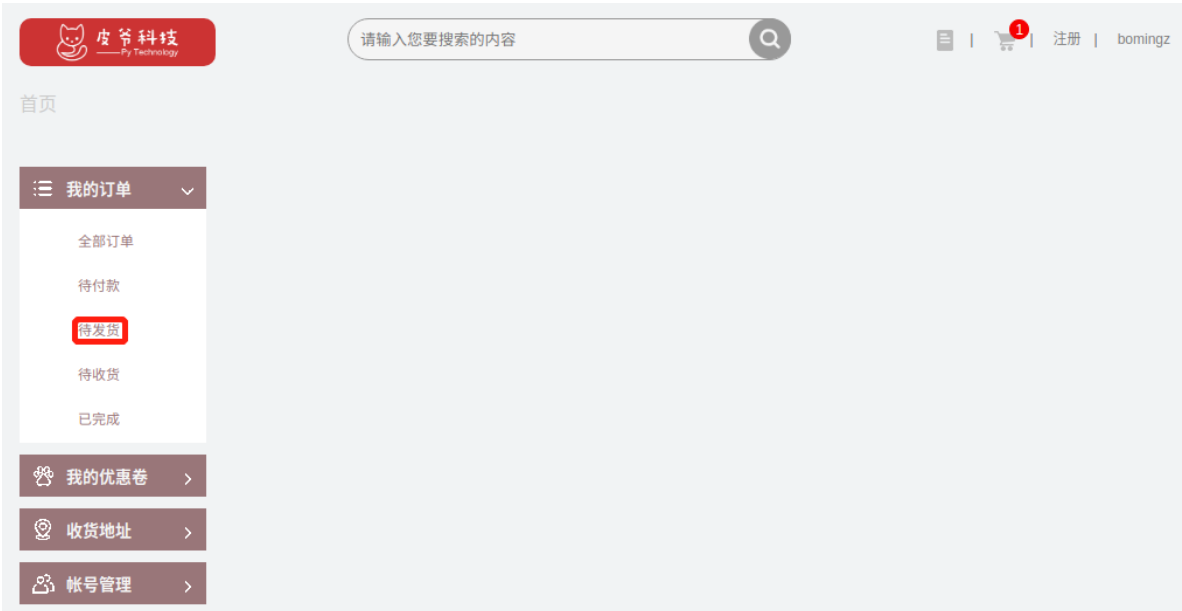


全部订单

订单页点击待付款按钮

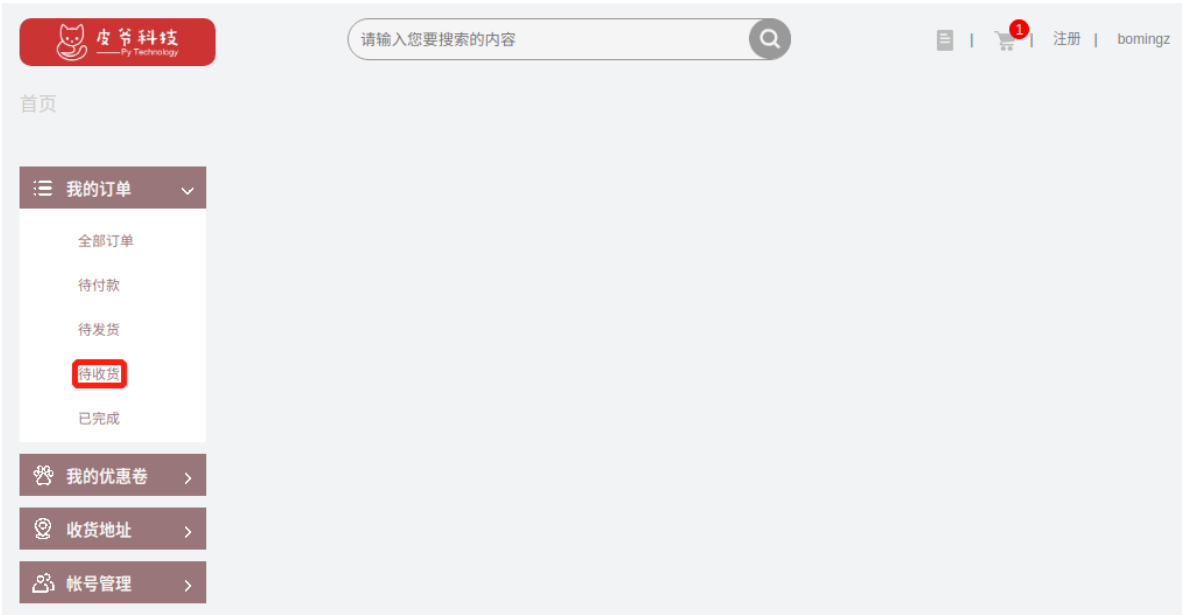
待付款

订单页点击带发货按钮



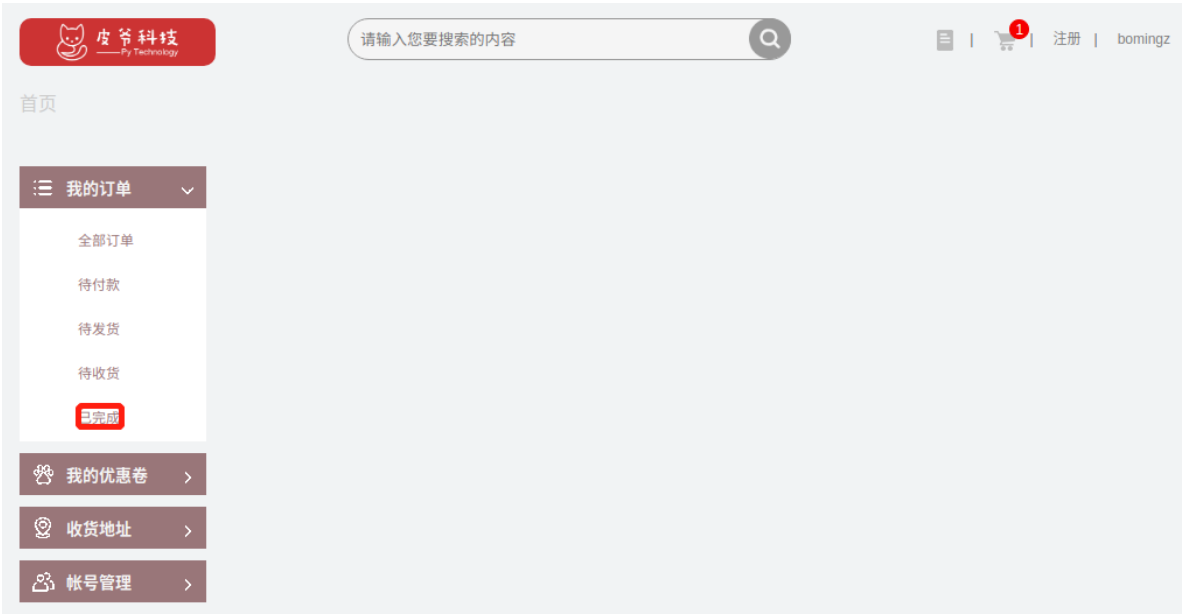
待发货

订单页点击待收货按钮



待收货

订单页点击已完成按钮



已完成

2.请求:

- **URL：** http://127.0.0.1:8000/v1/orders/processing/
- **请求方式:**GET
- **请求参数:**

参数	类型	是否必须	说明
status	int	是	业务类型码
order_status	str	是	订单状态

- **请求示例:**

```
127.0.0.1:8000/v1/orders/processing/?status=1&order_status=0
```

3.响应:

- **返回值类型:**JSON
- **响应内容:**

字段	含义	类型	备注
code	状态码	int	默认正常为200，异常见状态码
data	具体数据	dict	与error二选一
error	错误信息	char	与data二选一

- **响应格式:**

```
{"code":200,"data":data}
```

- **data参数信息:**

参数	类型	是否必须	说明
orders_list	list	是	订单信息列表

- **data数据示例:**


```
   "data":{"orders_list":[
              {"order_id":"2019111811504601",
               "order_total_count":1,
               "order_total_amount":"209.00",
               "order_freight":"1.00",
               "address":{
                   "title":"家",
                   "address":"北京市北京市市辖区东城区珍贝大厦",
                   "mobile":"13691433520",
                   "receiver":"习瓜瓜"},
               "status":1,
               "order_sku":[
                      {"id":4,
                       "default_image_url":"2_94OnDrI.jpg",
                       "name":"红袖添香",
                       "price":"199.00",
                       "count":1,
                       "total_amount":"199.00",
                       "sku_sale_attr_names":[
                           "颜色",
                           "尺寸"],
                       "sku_sale_attr_vals":[
                           "红色",
                           "15寸" ]}],
               "order_time":"2019-11-18 11:50:46"}]}
```


- **error参数信息:**

参数	类型	是否必须	说明
error_msg	str	是	错误信息说明

- **error数据示例:**

```
[{
    "error_msg":""
}]
```

- **状态码参考:**

状态码	响应信息
200	正常

4.代码展示

前端

```
var order_status = arguments[0] ? arguments[0] : 0;
        var status = arguments[1] ? arguments[1] : 1;
        var token=localStorage.getItem("dashop_token")
        $.ajax({
            type: "get",
            beforeSend:function(request){
                            request.setRequestHeader("authorization",token)
            },
            url: baseUrl+'/v1/orders/processing/',
            datatype: JSON,
            data:{"status":status,'order_status':order_status},
            success: function (response) {
                // 渲染响应回来的数据
            }
```

后端

```
elif status == 1:
    # 获取状态码为依据返回制定状态的订单.
    order_status = request.GET.get('order_status','')
    if order_status == '0':
        order_list = OrderInfo.objects.filter(user=user)
    else:
        order_list = OrderInfo.objects.filter(user=user,status=order_status)
    orders_lists = []
    for order in order_list:
        good_skus = OrderGoods.objects.filter(order=order)
        sku_list = []
        # 1.组织订单中sku数据
        for good_sku in good_skus:
            sku = good_sku.sku
            sku_sale_attr_names, sku_sale_attr_vals,_ = self.get_sku_list(sku)
            sku_list.append({
                'id': sku.id,
                'default_image_url': str(sku.default_image_url),
                'name': sku.name,
                'price': sku.price,
                'count': good_sku.count,
                'total_amount': sku.price * good_sku.count,
                "sku_sale_attr_names": sku_sale_attr_names,
                "sku_sale_attr_vals": sku_sale_attr_vals,
```

```
            })
        # 2.组织订单信息
        orders_lists.append({
            "order_id": order.order_id,
            "order_total_count": order.total_count,
            "order_total_amount": order.total_amount,
            "order_freight": order.freight,
            "address": {
                "title": order.address.tag,
                "address": order.address.address,
                "mobile": order.address.receiver_mobile,
                "receiver": order.address.receiver
            },
            "status": order.status,
            "order_sku": sku_list,
            "order_time": str(order.created_time)[0:19]
        })
    data = {
        'orders_list': orders_lists,
    }
    return http.JsonResponse({"code": 200,"data": data, 'base_url':PIC_URL})
```

4.确认收货API

1.事件触发

在登录状态下才可使用此功能

用户在订单页点击确认收货按钮

页面展示

点击确定后展示

2请求:

- **URL**：http://127.0.0.1:8000/v1/orders/processing/
- **请求方式:**GET
- **请求参数:**

参数	类型	是否必须	说明
status	int	是	业务类型码
order_id	str	是	订单状态

- **请求示例:**

```
127.0.0.1:8000/v1/orders/processing/?status=2
```

3.响应:

- **返回值类型:**JSON

- **响应内容:**

字段	含义	类型	备注
code	状态码	int	默认正常为200，异常见状态码

- **响应格式:**

```
{"code":200}
```

4.代码展示:

前端

```
function confirm(order_id){
        var token=localStorage.getItem("dashop_token")
        $.ajax({
            url:baseUrl+'/v1/orders/processing/',
            data:{status:2,order_id:order_id},
            datatype:JSON,
            type:'get',
            beforeSend:function(request){
                            request.setRequestHeader("authorization",token)
            },
            success:function (data) {
                if(data.code==200){
                    alert('确认收货成功')
                    getData(4)
                }
            },
            error:function (e) {
                alert('system error!')
            }
        })
    }
```

后端

```
elif status == 2:
    order_id = request.GET.get("order_id")
    order = OrderInfo.objects.filter(order_id=order_id)[0]
    order.status = 4
    order.save()
    return http.JsonResponse({"code": 200, 'base_url':PIC_URL})
```

5.订单支付API

1.事件定义

登录状态才可使用此功能

用户在订单页点击去付款按钮

订单支付页响应数据

点击确认支付按钮

支付宝登录

输入支付宝支付密码

支付宝响应

dadashop响应

2.请求

- **URL：** http://127.0.0.1:8000/v1/orders/processing/
- **请求方式:**POST
- **请求参数:**

参数	类型	是否必须	说明
status	int	是	业务类型码
order_id	str	是	订单id

- **请求示例:**

```
http://127.0.0.1:8000/v1/orders/processing/?status=1&order_id=2019111811504601
```

3.响应:

- **返回值类型:**JSON
- **响应内容:**

字段	含义	类型	备注
code	状态码	int	默认正常为200，异常见状态码
data	具体数据	dict	与error二选一
error	错误信息	char	与data二选一

- **响应格式:**

```
{"code":200,"data":data}
```

- **data参数信息:**

参数	类型	是否必须	说明

参数	类型	是否必须	说明
saller	str	是	商家名称
total_amount	decimail	是	总价格
order_id	str	是	订单号
pay_url	str	是	第三方支付路由

- **data数据示例:**

```
    [{'saller': '达达商城',
      'total_amount': total_amount + 10,
      'order_id': order_id,
      'pay_url': pay_url
       }]
```

- **error参数信息:**

参数	类型	是否必须	说明
error_msg	str	是	错误信息说明

- **error数据示例:**

```
[{
    "error_msg":""
}]
```

- **状态码参考:**

状态码	响应信息
200	正常

4.代码展示

前端

```
$.ajax({
        type: 'post',
        beforeSend:function(request){
            request.setRequestHeader("authorization",token)
        },
        url: baseUrl+'/v1/orders/processing/',
        data:JSON.stringify({
            "status":1,
            "order_id":order_id
        }),
        datatype: 'json',
        async: false,
        success: function (response) {
            var result = JSON.stringify(response)
            var results = JSON.parse(result)
            $("#total_amount").html(results.data.total_amount-10+"元")
```

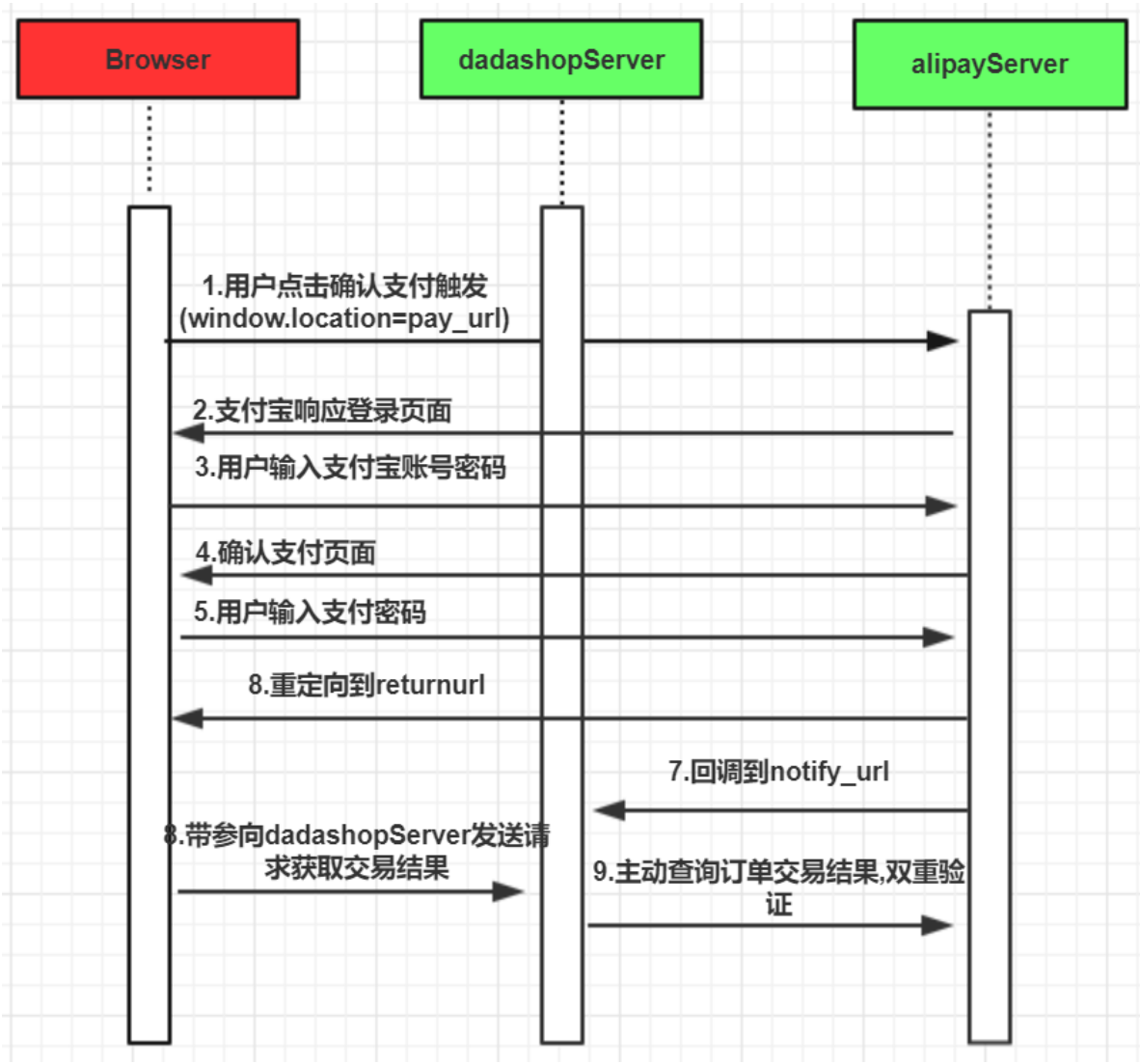
```
            $("#order_id").html("支付订单:"+results.data.order_id)
            $("#saller").html("收款方:"+results.data.saller+"  ")
            $(".pay_leftTime").html("总金额(含运费):"+results.data.total_amount)
            var pay_url = results.data.pay_url
            $(function(){
                $("#pay_now").click(function () {
                    window.location = pay_url
                })
            })
        }
})
```

后端

```
 elif status == 1:
    order_id = json.loads(request.body.decode()).get("order_id")
    order = OrderInfo.objects.filter(order_id=order_id)[0]
    # 1.生成支付链接
    sku_goods = OrderGoods.objects.filter(order=order_id)
    order_string =
self.get_order_string(order.order_id,order.total_amount,sku_goods)
    # 构建让用户跳转的支付链接
    pay_url = "https://openapi.alipaydev.com/gateway.do?" + order_string
    # 2.组织数据
    data = {
        'saller': '达达商城',
        'total_amount': order.total_amount+10,
        'order_id': order_id,
        'pay_url': pay_url
    }
    # 响应
    return http.JsonResponse({'code': 200, 'data': data})
```

6.第三方支付接口-主动查询

支付宝业务处理时序图



1.请求

- **URL：** http://127.0.0.1:8000/v1/orders/success/
- **请求方式:**GET
- **请求参数:**

参数	类型	是否必须	说明
out_trade_no	str	是	订单号
total_amount	decimal	是	订单金额
sign	str	是	密钥

- **请求示例:**

```
http://127.0.0.1:8000/v1/orders/success/
```

2.响应:

- **返回值类型:**JSON
- **响应内容:**

字段	类型	备注
code	int	响应码
data	dict	支付状态

- **响应格式:**

```
{"code":200,"data":data}
```

- **data参数内容:**

参数	类型	是否必须	说明
order_id	str	是	订单编号
total_amount	int	是	商品价格

3.代码展示:

```
 # 1.获取参数字典,验签结果,alipay对象
success_dict,ali_verify,alipay = self.get_sdict_ali_verify(request,1)
# 2.根据验证结果进行业务处理
if ali_verify is True:
    order_id = success_dict.get('out_trade_no',None)
    order = OrderInfo.objects.filter(order_id=order_id)[0]
    total_amount = success_dict.get('total_amount', None)
    if order.status == 2:
        data = {
            "order_id": order_id,
            "total_amount": total_amount
        }
        return http.JsonResponse({"code": 200, "data": data})
    # 主动查询
    else:
        result = alipay.api_alipay_trade_query(out_trade_no=order_id)
        print("主动查询")
        if result.get("trade_status","") == "TRADE_SUCCESS":
            order = OrderInfo.objects.filter(order_id=order_id)[0]
            order.status = 2
            order.save()
            data = {
                "order_id": order_id,
                "total_amount": total_amount
            }
            return http.JsonResponse({"code": 200, "data": data})
        else:
            data = {
                "order_id": order_id,
                "total_amount": total_amount
            }
            return http.JsonResponse({"code": 50105, "data": data})
else:
    return HttpResponse("非法访问")
```

7.第三方支付接口-支付宝回调

1.请求

- **URL：** http://127.0.0.1:8000/v1/orders/success/
- **请求方式:**GET
- **请求参数:**

参数	类型	是否必须	说明
out_trade_no	str	是	订单号
trade_status	str	是	订单支付状态
sign	str	是	密钥

2.响应

- **返回值类型:**字符串
- **响应内容:**"success"

3.代码展示

```
def post(self,request):
        """
        处理支付宝的付款回调业务
        :param request:
        :return:
        """
        # 1.获取参数字典,验签结果,alipay对象
        success_dict, ali_verify, alipay = self.get_sdict_ali_verify(request, 2)
        # 2.根据验证结果进行业务处理
        if ali_verify is True:
            trade_status = success_dict.get('trade_status',None)
            if trade_status == "TRADE_SUCCESS":
                order_id = success_dict.get('out_trade_no',None)
                order = OrderInfo.objects.filter(order_id=order_id)[0]
                order.status = 2
                order.save()
                return HttpResponse("seccess")
        else:
            return HttpResponse("非法访问")
```