

尚品汇商城

一、支付宝介绍

1.1 支付宝简介

[支付宝（中国）网络技术有限公司](#)^[1] 是国内的第三方支付平台，致力于提供“简单、安全、快速”的支付解决方案^[2]。支付宝公司从 2004 年建立开始，始终以“信任”作为产品和服务的核心。旗下有“支付宝”与“支付宝钱包”两个独立品牌。自 2014 年第二季度开始成为当前全球最大的[移动支付](#)厂商。

当用户提交订单会跳转到选择支付渠道页面！

<http://payment.gmall.com/pay.html?orderId=43>

✓ 订单提交成功，请您及时付款，以便尽快为您发货~~

请您在提交订单4小时之内完成支付，超时订单会自动取消。订单号：145687

应付金额：¥17,654

重要说明：

1. 品优购商城支付平台目前支持[支付宝](#)支付方式。
2. 其它支付渠道正在调试中，敬请期待。
3. 为了保证您的购物支付流程顺利完成，请保存以下支付宝信息。

支付宝账户信息：（很重要，请保存！！）

- 支付帐号：duqthf1038@sandbox.com
- 密码：111111
- 支付密码：111111

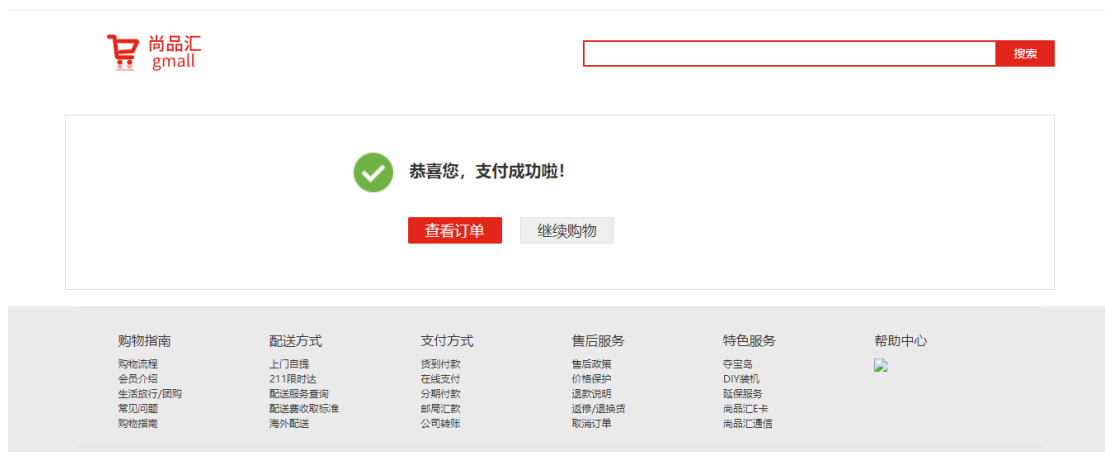
支付平台



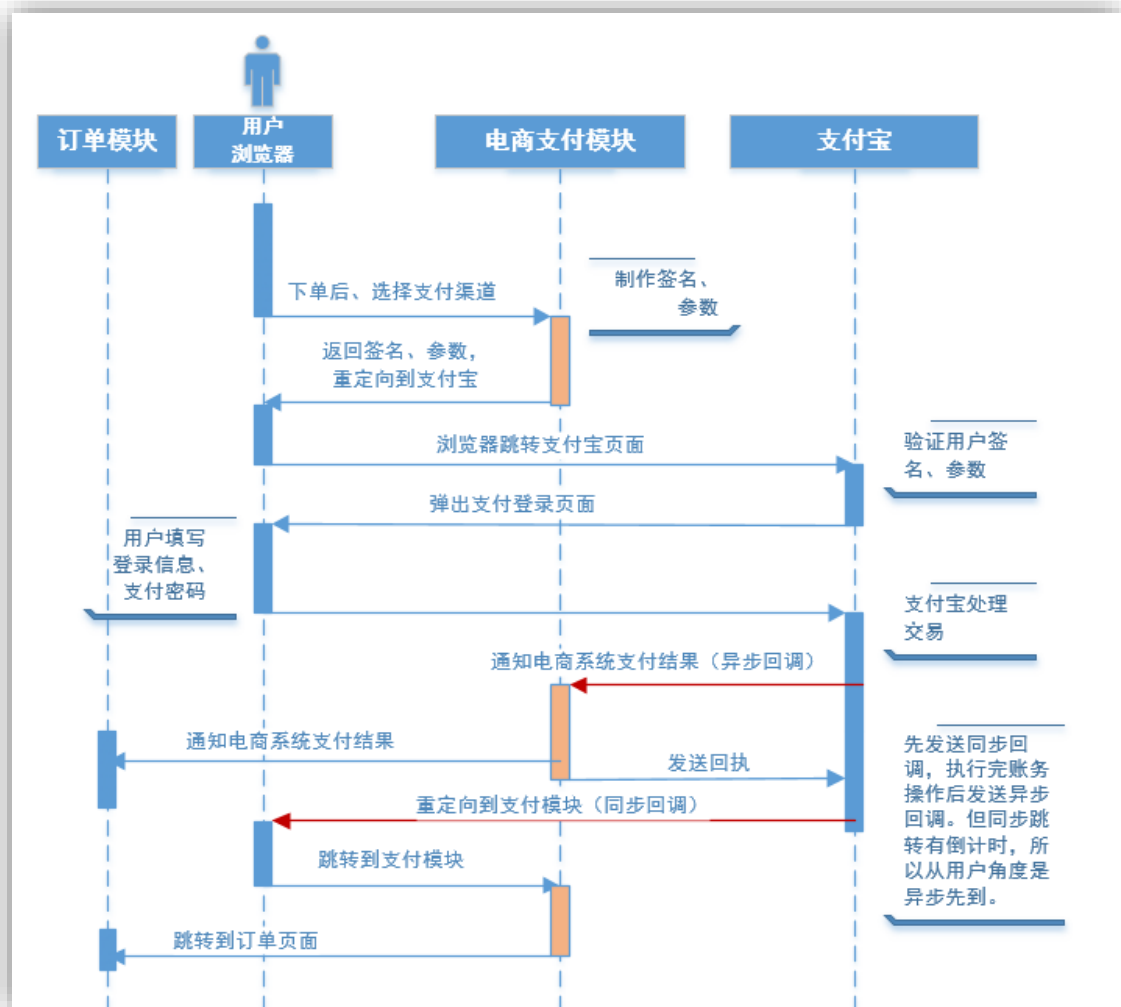
当用户点击立即支付时生成支付的二维码



使用支付宝 app 进行扫码支付



1.2 过程分析



1.3 对接支付宝的准备工作

1、申请条件

1. 企业或个体工商户可申请；
2. 提供真实有效的营业执照，且支付宝账户名称需与营业执照主体一致；
3. 网站能正常访问且页面信息有完整商品内容；
4. 网站必须通过 ICP 备案，个体户备案需与账户主体一致。

(团购类网站不支持个体工商户签约)

支付手续费

电脑网站支付

服务名称	费率	服务期限
单笔费率	0.6%	1年

费率说明：

助力中小商户，从签约日至2018.12.31日优惠费率为0.55%（不包括特殊行业）

特殊行业：休闲游戏；网络游戏点卡、游戏渠道代理；游戏系统商；网游周边服务、交易平台；网游运营商（含网页游戏）

1.4 申请步骤：

- 1、支付宝商家中心中申请 <https://www.alipay.com/>
- 2、<https://b.alipay.com/signing/productSetV2.htm?mrchportalwebServer=https%3A%2F%2Fmrchportalweb.alipay.com>



一个工作日后登录到蚂蚁金服开发者中心：



可以查看到一个已经签约上线的应用。其中非常重要是这个 **APPID**，需要记录下来之后的程序中要用到这个参数。

点击查看



到此为止，电商网站可以访问支付宝的最基本的准备已经完成。

接下来搭建支付模块

二、支付模块搭建

支付宝开发手册: <https://docs.open.alipay.com/270/105900/>

2.1 搭建 service-payment

2.1.1 搭建 service-payment

搭建方式如 service-order

2.1.2 修改 pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
    https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>com.atguigu.gmall</groupId>
    <artifactId>service</artifactId>
    <version>1.0</version>
  </parent>

  <version>1.0</version>
  <artifactId>service-payment</artifactId>
  <packaging>jar</packaging>
  <name>service-payment</name>
  <description>service-payment</description>

  <dependencies>

    <dependency>
      <groupId>com.atguigu.gmall</groupId>
      <artifactId>service-order-client</artifactId>
      <version>1.0</version>
    </dependency>

  </dependencies>
```

```
<build>
  <finalName>service-payment</finalName>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
    </plugin>
  </plugins>
</build>

</project>
```

2.1.3 添加配置

bootstrap.properties

```
spring.application.name=service-payment
spring.profiles.active=dev
spring.cloud.nacos.discovery.server-addr=192.168.200.129:8848
spring.cloud.nacos.config.server-addr=192.168.200.129:8848
spring.cloud.nacos.config.prefix=${spring.application.name}
spring.cloud.nacos.config.file-extension=yaml
spring.cloud.nacos.config.shared-configs[0].data-id=common.yaml
```

2.1.4 启动类

```
package com.atguigu.gmall.payment;

@SpringBootApplication
@ComponentScan({"com.atguigu.gmall"})
@EnableDiscoveryClient
@EnableFeignClients(basePackages= {"com.atguigu.gmall"})
public class ServicePaymentApplication {

    public static void main(String[] args) {
        SpringApplication.run(ServicePaymentApplication.class, args);
    }

}
```

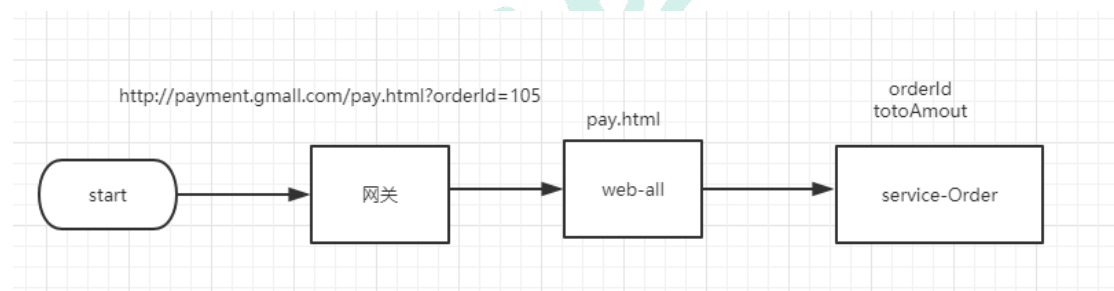
2.2 导入 sdk 包

<https://mvnrepository.com/artifact/com.alipay.sdk/alipay-sdk-java>

```
<!-- 导入支付宝支付 sdk -->
<dependency>
  <groupId>com.alipay.sdk</groupId>
  <artifactId>alipay-sdk-java</artifactId>
  <version>4.8.73.ALL</version>
</dependency>
```

三、显示付款页面信息

支付页面信息展示流程



3.1 在根据订单 id 获取订单信息接口

3.1.1 在 service-order 添加接口

OrderService 接口

```
/**
 * 根据订单 Id 查询订单信息
 * @param orderId
 * @return
 */
```



```
OrderInfo getOrderInfo(Long orderId);
```

实现类

```
@Override
public OrderInfo getOrderInfo(Long orderId) {
    OrderInfo orderInfo = orderInfoMapper.selectById(orderId);
    QueryWrapper<OrderDetail> queryWrapper = new QueryWrapper<>();
    queryWrapper.eq("order_id", orderId);
    List<OrderDetail> orderDetailList =
    orderDetailMapper.selectList(queryWrapper);
    orderInfo.setOrderDetailList(orderDetailList);
    return orderInfo;
}
```

OrderApiController

```
/**
 * 内部调用获取订单
 * @param orderId
 * @return
 */
@GetMapping("inner/getOrderInfo/{orderId}")
public OrderInfo getOrderInfo(@PathVariable(value = "orderId") Long
orderId){
    return orderService.getOrderInfo(orderId);
}
```

3.1.2 在 service-order-client 添加接口

OrderFeignClient

```
/**
 * 获取订单
 * @param orderId
 * @return
 */
@GetMapping("/api/order/inner/getOrderInfo/{orderId}")
OrderInfo getOrderInfo(@PathVariable(value = "orderId") Long
orderId);
```

OrderDegradeFeignClient

```
@Override
public OrderInfo getOrderInfo(Long orderId) {
    return null;
}
```

3.2 server-gateway 模块网关配置

```
- id: web-payment
  uri: lb://web-all
  predicates:
    - Host=payment.gmall.com
- id: service-payment
  uri: lb://service-payment
  predicates:
    - Path=/*/payment/** # 路径匹配
```

3.3 创建支付控制器 PaymentController

Web-all 模块

```
package com.atguigu.gmall.payment.controller;
@Controller
public class PaymentController {

    @Autowired
    private OrderFeignClient orderFeignClient;

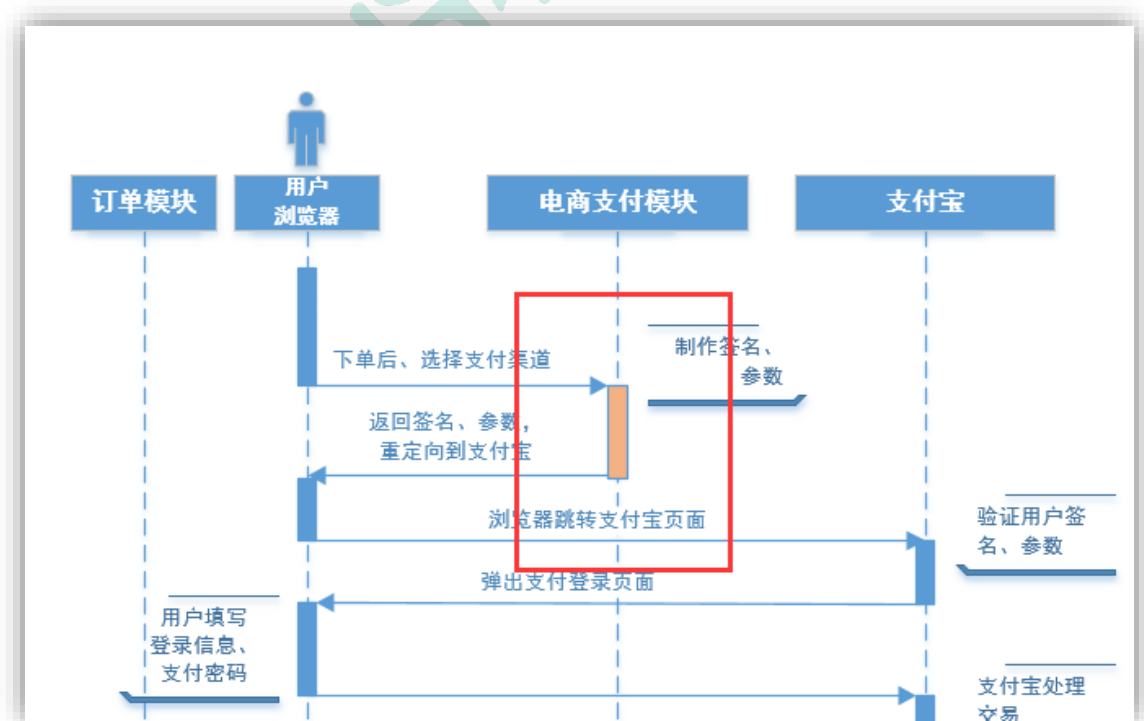
    /**
     * 支付页
     * @param request
     * @return
     */
    @GetMapping("pay.html")
    public String success(HttpServletRequest request, Model model) {
        String orderId = request.getParameter("orderId");
        OrderInfo orderInfo =
        orderFeignClient.getOrderInfo(Long.parseLong(orderId));
        model.addAttribute("orderInfo", orderInfo);
        return "payment/pay";
    }
}
```

3.4 页面渲染

页面资源： \templates\payment\pay.html

```
<div class="checkout-tit">
  <h4 class="tit-txt"><span class="success-icon"></span><span
class="success-info">订单提交成功，请您及时付款，以便尽快为您发货
~~</span></h4>
  <div class="paymark">
    <span class="f1">请您在提交订单<em class="orange time">4 小时
</em>之内完成支付，超时订单会自动取消。订单号：<em
th:text="${orderInfo.id}">00</em></span>
    <span class="fr"><em class="sui-lead">应付金额：</em><em
class="orange money" th:text="' ¥ '+${orderInfo.totalAmount}">
¥1</em></span>
  </div>
</div>
```

四、支付功能实现



支付宝有了同步通知为什么还需要异步通知？

同步回调两个作用

第一是从支付宝的页面上返回自己的网站继续后续操作；
第二是携带支付状态的 get 参数；让自己的网站用于验证；

同步通知后，还需要异步通知主要是为了防止出现意外情况；

因为涉及到金钱；这是一个对安全和稳定要求比较严格的场景；

如果同步通知的过程中；用户不小心关闭了浏览器；或者浏览器卡死了；

异步也能收到通知；记录支付状态；

即便是用户端没问题；万一自己的服务器网络异常了一下呢？

如果自己的服务器没有正确返回接受到通知的状态；

支付宝的服务器会在一段时间内持续的往自己的服务器发送**异步通知**；
一直到成功；

顺便去确认了下；这个一段时间是：

25 小时以内完成 8 次通知（通知的间隔频率一般是：4m, 10m, 10m, 1h, 2h, 6h, 15h）

4.1 思路分析



1. 将支付数据保存到数据库，以便跟支付宝进行对账
2. 生成要支付的二维码

生成二维码需要的参数列表请参考官方文档

<https://opendocs.alipay.com/open/270/105899>

4.2 保存支付信息的表结构

表结构 payment_info

payment_info(支付信息表)	
 id	编号
 out_trade_no	对外业务编号
 order_id	订单编号
 payment_type	支付类型 (微信 支付宝)
 trade_no	交易编号
 total_amount	支付金额
 subject	交易内容
 payment_status	支付状态
 create_time	创建时间
 callback_time	回调时间
 callback_content	回调信息

id	主键自动生成
out_trade_no	订单中已生成的对外交易编号。订单中获取
order_id	订单编号
payment_type	支付类型 (微信与支付宝)
trade_no	交易号, 回调时生成
total_amount	订单金额。订单中获取
subject	交易内容。利用商品名称拼接。
payment_status	支付状态, 默认值未支付。
create_time	创建时间, 当前时间。
callback_time	回调时间, 初始为空, 支付宝异步回调时记录
callback_content	回调信息, 初始为空, 支付宝异步回调时记录

4.3 编写接口, 实现类 PaymentService

4.3.1 接口 PaymentService

```
package com.atguigu.gmall.payment.service;

import com.atguigu.gmall.model.order.OrderInfo;
```

```
public interface PaymentService {  
    /**  
     * 保存交易记录  
     * @param orderInfo  
     * @param paymentType 支付类型 (1: 微信 2: 支付宝)  
     */  
    void savePaymentInfo(OrderInfo orderInfo, String paymentType);  
}
```

4.3.2 定义 PaymentMapper 接口

PaymentInfoMapper

```
package com.atguigu.gmall.payment.mapper;  
  
@Mapper  
public interface PaymentInfoMapper extends BaseMapper<PaymentInfo> {  
}
```

4.3.3 实现类 PaymentServiceImpl

```
package com.atguigu.gmall.payment.service.impl;  
  
@Service  
public class PaymentServiceImpl implements PaymentService {  
  
    @Autowired  
    private PaymentInfoMapper paymentInfoMapper;  
  
    @Override  
    public void savePaymentInfo(OrderInfo orderInfo, String paymentType) {  
        QueryWrapper<PaymentInfo> queryWrapper = new QueryWrapper<>();  
        queryWrapper.eq("order_id", orderInfo.getId());  
        queryWrapper.eq("payment_type", paymentType);  
        Integer count = paymentInfoMapper.selectCount(queryWrapper);  
        if(count > 0) return;  
  
        // 保存交易记录  
        PaymentInfo paymentInfo = new PaymentInfo();  
        paymentInfo.setCreateTime(new Date());  
    }  
}
```

```
        paymentInfo.setOrderId(orderInfo.getId());
        paymentInfo.setPaymentType(paymentType);
        paymentInfo.setOutTradeNo(orderInfo.getOutTradeNo());
        paymentInfo.setPaymentStatus(PaymentStatus.UNPAID.name());
        paymentInfo.setSubject(orderInfo.getTradeBody());
        //paymentInfo.setSubject("test");
        paymentInfo.setTotalAmount(orderInfo.getTotalAmount());

        paymentInfoMapper.insert(paymentInfo);
    }
}
```

4.4 编写支付宝支付接口

4.4.1 制作 AlipayClient 工具类

创建配置类

```
package com.atguigu.gmall.payment.config;

@Configuration
public class AlipayConfig {

    @Value("${alipay_url}")
    private String alipay_url;

    @Value("${app_private_key}")
    private String app_private_key;

    @Value("${app_id}")
    private String app_id;

    public final static String format="json";
    public final static String charset="utf-8";
    public final static String sign_type="RSA2";

    public static String return_payment_url;

    public static String notify_payment_url;

    public static String return_order_url;

    public static String alipay_public_key;
```

```
@Value("${alipay_public_key}")
public void setAlipay_public_key(String alipay_public_key) {
    AlipayConfig.alipay_public_key = alipay_public_key;
}

@Value("${return_payment_url}")
public void setReturn_url(String return_payment_url) {
    AlipayConfig.return_payment_url = return_payment_url;
}

@Value("${notify_payment_url}")
public void setNotify_url(String notify_payment_url) {
    AlipayConfig.notify_payment_url = notify_payment_url;
}

@Value("${return_order_url}")
public void setReturn_order_url(String return_order_url) {
    AlipayConfig.return_order_url = return_order_url;
}

// <bean id="alipayClient" class="com.alipay.api.AlipayClient">
</bean>
@Bean
public AlipayClient alipayClient(){
    //      AlipayClient      alipayClient      =      new
DefaultAlipayClient("https://openapi.alipay.com/gateway.do", APP_ID,
APP_PRIVATE_KEY, FORMAT, CHARSET, ALIPAY_PUBLIC_KEY, SIGN_TYPE); //
获得初始化的AlipayClient
    AlipayClient      alipayClient=new
DefaultAlipayClient(alipay_url,app_id,app_private_key,format,charset
, alipay_public_key,sign_type );
    return alipayClient;
}
}
```

4.4.2 编写支付宝下单

4.4.2.1 接口

```
package com.atguigu.gmall.payment.service;

public interface AlipayService {

    String createaliPay(Long orderId);
}
```



```
}
```

4.4.2.2 实现类

```
package com.atguigu.gmall.payment.service.impl;

@Service
public class AlipayServiceImpl implements AlipayService {

    @Autowired
    private AlipayClient alipayClient;

    @Autowired
    private OrderFeignClient orderFeignClient;

    @Autowired
    private PaymentService paymentService;

    @Override
    public String createaliPay(Long orderId) {
        // 根据订单Id 获取orderInfo
        OrderInfo orderInfo =
        orderFeignClient.getOrderInfo(orderId);
        if ("PAID".equals(orderInfo.getOrderStatus()) ||
        "CLOSED".equals(orderInfo.getOrderStatus())){
            return "该订单已经完成或已经关闭!";
        }
        // 调用保存交易记录方法!
        paymentService.savePaymentInfo(orderInfo,
        PaymentType.ALIPAY.name());

        String form = "";
        // 创建 AlipayClient 对象! AlipayClient 这个对象注入到
        spring 容器中!
        // AlipayClient alipayClient = new
        DefaultAlipayClient("https://openapi.alipay.com/gateway.do",
        APP_ID, APP_PRIVATE_KEY, FORMAT, CHARSET, ALIPAY_PUBLIC_KEY,
        SIGN_TYPE); //获得初始化的AlipayClient
        AlipayTradePagePayRequest alipayRequest = new
        AlipayTradePagePayRequest(); //创建API 对应的request
        // 同 步 回 调
        http://api.gmall.com/api/payment/alipay/callback/return
        alipayRequest.setReturnUrl(AlipayConfig.return_payment_url);
        // 异步回调
        alipayRequest.setNotifyUrl("http://domain.com/CallBack/notify_url.j
```

```
sp" ); // 在公共参数中设置回跳和通知地址
// 封装业务参数
HashMap<String, Object> map = new HashMap<>();
// 第三方业务编号!
map.put("out_trade_no",orderInfo.getOutTradeNo());
map.put("product_code","FAST_INSTANT_TRADE_PAY");
map.put("total_amount","0.01");
map.put("subject",orderInfo.getTradeBody());
// 设置二维码过期时间
map.put("timeout_express","10m");
alipayRequest.setBizContent(JSON.toJSONString(map));
try {
    form
    alipayClient.pageExecute(alipayRequest).getBody(); //调用 SDK 生成表单
} catch (AlipayApiException e) {
    e.printStackTrace();
}
return form;
}
```

4.4.2.3 控制器

```
AlipayController

@Controller
@RequestMapping("/api/payment/alipay")
public class AlipayController {

    @Autowired
    private AlipayService alipayService;

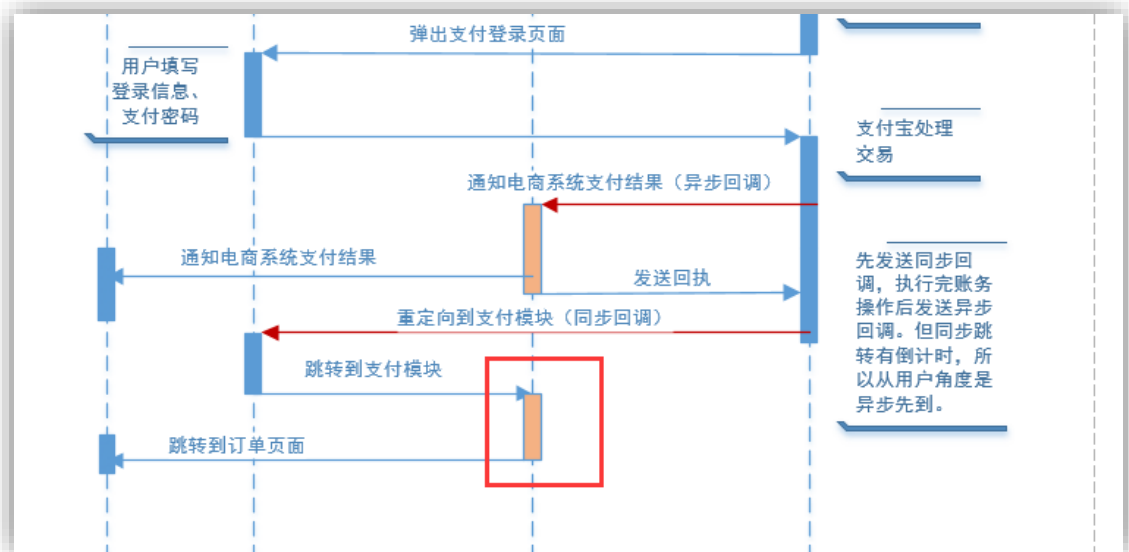
    @RequestMapping("submit/{orderId}")
    @ResponseBody
    public String submitOrder(@PathVariable Long orderId){
        String form = alipayService.createaliPay(orderId);
        return form;
    }
}
```

4.4.3 前端页面

```
<ul class="payType">
    <a
    th:href="@{http://api.gmall.com/api/payment/alipay/submit/{orderId}}(>
```

```
orderId=${orderInfo.id}}" target="_blank"><li></li></a>
</ul>
```

五、支付后回调—同步回调



5.1 控制器 AlipayController

```
/**
 * 支付宝回调
 * @return
 */
@RequestMapping("callback/return")
public String callBack() {
    // 同步回调给用户展示信息
    return "redirect:" + AlipayConfig.return_order_url;
}
```

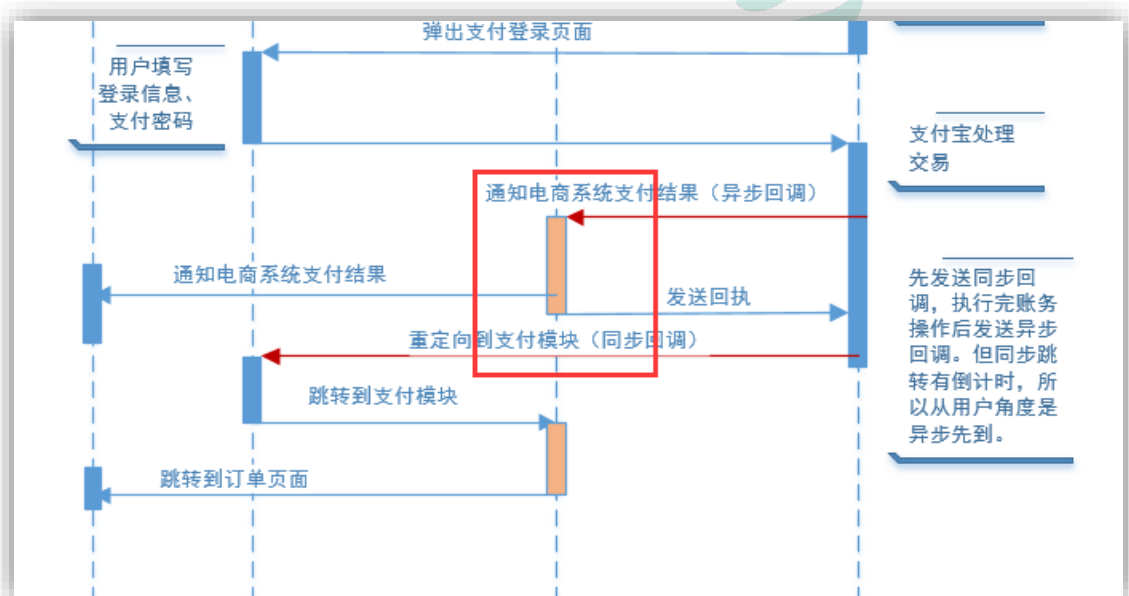
5.2 在 web-all 项目中添加对应的返回控制器

```
/**
```

```

* 支付成功页
* @return
*/
@GetMapping("pay/success.html")
public String success() {
    return "payment/success";
}
    
```

六、支付宝回调—异步回调



异步回调有两个重要的职责：

确认并记录用户已付款，通知电商模块。新版本的支付接口已经取消了同步回调的支付结果传递。所以用户付款成功与否全看异步回调。

接收到回调要做的事情：

- 1、验证回调信息的真伪
- 2、验证用户付款的成功与否
- 3、把新的支付状态写入支付信息表{paymentInfo}中。

4、通知电商

5、给支付宝返回回执。

6.1 控制器 AlipayController

```
@Autowired
private PaymentService paymentService;

@PostMapping("/callback/notify")
@ResponseBody
public String callbackNotify(@RequestParam Map<String, String>
paramsMap){
    System.out.println("你回来了.....");
    // Map<String, String> paramsMap = ... // 将异步通知中收到的所有参数都
    存放到map 中
    boolean signVerified = false; //调用 SDK 验证签名
    try {
        signVerified = AlipaySignature.rsaCheckV1(paramsMap,
        AlipayConfig.alipay_public_key,
        AlipayConfig.charset,
        AlipayConfig.sign_type);
    } catch (AlipayApiException e) {
        e.printStackTrace();
    }
    // 获取异步通知的参数中的订单号!
    String outTradeNo = paramsMap.get("out_trade_no");
    // 获取异步通知的参数中的订单总金额!
    String totalAmount = paramsMap.get("total_amount");
    // 获取异步通知的参数中的 appId!
    String appId = paramsMap.get("app_id");
    // 获取异步通知的参数中的交易状态!
    String tradeStatus = paramsMap.get("trade_status");
    // 根据 outTradeNo 查询数据!
    PaymentInfo paymentInfo = paymentInfo
    this.paymentService.getPaymentInfo(outTradeNo,
    PaymentType.ALIPAY.name());
    // 保证异步通知的幂等性! notify_id
    String notifyId = paramsMap.get("notify_id");

    // true:
    if(signVerified){
        // TODO 验签成功后, 按照支付结果异步通知中的描述, 对支付结果中的业务
        内容进行二次校验, 校验成功后在 response 中返回 success 并继续商户自身业务处理,
        校验失败返回 failure
        if (paymentInfo==null || new BigDecimal("0.01").compareTo(new
        BigDecimal(totalAmount))!=0
            || !appId.equals(app_id)){
            return "failure";
        }
    }
}
```

```
// 放入 redis! setnx: 当 key 不存在的时候生效!
Boolean flag =
this.redisTemplate.opsForValue().setIfAbsent(notifyId, notifyId, 1462,
TimeUnit.MINUTES);
// 说明已经处理过了!
if (!flag){
    return "failure";
}
if ("TRADE_SUCCESS".equals(tradeStatus) ||
"TRADE_FINISHED".equals(tradeStatus)){
    // 修改交易记录状态! 再订单状态!

this.paymentService.paySuccess(outTradeNo, PaymentType.ALIPAY.name(), par
amsMap);
//
this.paymentService.paySuccess(paymentinfo.getId(), paramsMap);
    return "success";
}
}else{
    // TODO 验签失败则记录异常日志, 并在 response 中返回 failure.
    return "failure";
}
return "failure";
}
```

6.2 接口 PaymentService

```
/**
 * 获取 paymentInfo 对象
 * @param outTradeNo
 * @param name
 * @return
 */
PaymentInfo getPaymentInfo(String outTradeNo, String name);

/**
 * 支付成功更新交易记录方法
 * @param outTradeNo
 * @param name
 * @param paramMap
 */
void paySuccess(String outTradeNo, String name, Map<String, String>
paramMap);

/**
 * 根据 outTradeNo 支付方式 name 更新数据
 * @param outTradeNo
 * @param name
 * @param paymentInfo
 */
```

```
void updatePaymentInfo(String outTradeNo, String name, PaymentInfo paymentInfo);
```

6.3 实现类

```
@Override
public PaymentInfo getPaymentInfo(String outTradeNo, String name) {
    // select * from payment_info where out_trade_no = ? and
    // payment_type = ?
    QueryWrapper<PaymentInfo> paymentInfoQueryWrapper = new
    QueryWrapper<>();
    paymentInfoQueryWrapper.eq("out_trade_no", outTradeNo);
    paymentInfoQueryWrapper.eq("payment_type", name);
    return paymentInfoMapper.selectOne(paymentInfoQueryWrapper);
}

@Override
public void paySuccess(String outTradeNo, String paymentType,
    Map<String, String> paramsMap) {

    // 根据 outTradeNo, paymentType 查询
    PaymentInfo paymentInfoQuery = this.getPaymentInfo(outTradeNo,
    paymentType);
    if (paymentInfoQuery == null) {
        return;
    }

    try {
        // 改造一下更新的方法!
        PaymentInfo paymentInfo = new PaymentInfo();
        paymentInfo.setCallbackTime(new Date());
        paymentInfo.setPaymentStatus(PaymentStatus.PAID.name());
        paymentInfo.setCallbackContent(paramsMap.toString());
        paymentInfo.setTradeNo(paramsMap.get("trade_no"));
        // 查询条件也可以作为更新条件!
        this.updatePaymentInfo(outTradeNo, paymentType, paymentInfo);
    } catch (Exception e) {
        // 删除 key
        this.redisTemplate.delete(paramsMap.get("notify_id"));
        e.printStackTrace();
    }
}

// 更新交易状态记录!
public void updatePaymentInfo(String outTradeNo, String name,
    PaymentInfo paymentInfo) {
    QueryWrapper<PaymentInfo> paymentInfoQueryWrapper = new
    QueryWrapper<>();
```

```
paymentInfoQueryWrapper.eq("out_trade_no", outTradeNo);
paymentInfoQueryWrapper.eq("payment_type", name);
paymentInfoMapper.update(paymentInfo, paymentInfoQueryWrapper);
}
```

七、退款

直接在浏览器发起请求即可！

商户与客户协商一致的情况下，才可以退款！

7.1 接口

```
boolean refund(Long orderId);
```

7.2 实现类

```
@Override
public boolean refund(Long orderId) {

    AlipayTradeRefundRequest request = new
    AlipayTradeRefundRequest();

    OrderInfo orderInfo = orderFeignClient.getOrderInfo(orderId);
    HashMap<String, Object> map = new HashMap<>();
    map.put("out_trade_no", orderInfo.getOutTradeNo());
    map.put("refund_amount", orderInfo.getTotalAmount());
    map.put("refund_reason", "颜色浅了点");
    // out_request_no

    request.setBizContent(JSON.toJSONString(map));
    AlipayTradeRefundResponse response = null;
    try {
        response = alipayClient.execute(request);
    } catch (AlipayApiException e) {
        e.printStackTrace();
    }
    if (response.isSuccess()) {
```



```
// 更新交易记录 : 关闭
PaymentInfo paymentInfo = new PaymentInfo();
paymentInfo.setPaymentStatus(PaymentStatus.CLOSED.name());
paymentService.updatePaymentInfo(orderInfo.getOutTradeNo(),
paymentInfo);
return true;
} else {
    return false;
}
}
```

7.3 控制器

```
// 发起退款 ! http://localhost:8205/api/payment/alipay/refund/20
@RequestMapping("refund/{orderId}")

@ResponseBody
public Result refund(@PathVariable(value = "orderId")Long orderId) {
    // 调用退款接口
    boolean flag = alipayService.refund(orderId);

    return Result.ok(flag);
}
```