谷粒商城

版本：V 1.1

支付宝接口

# 一、支付业务

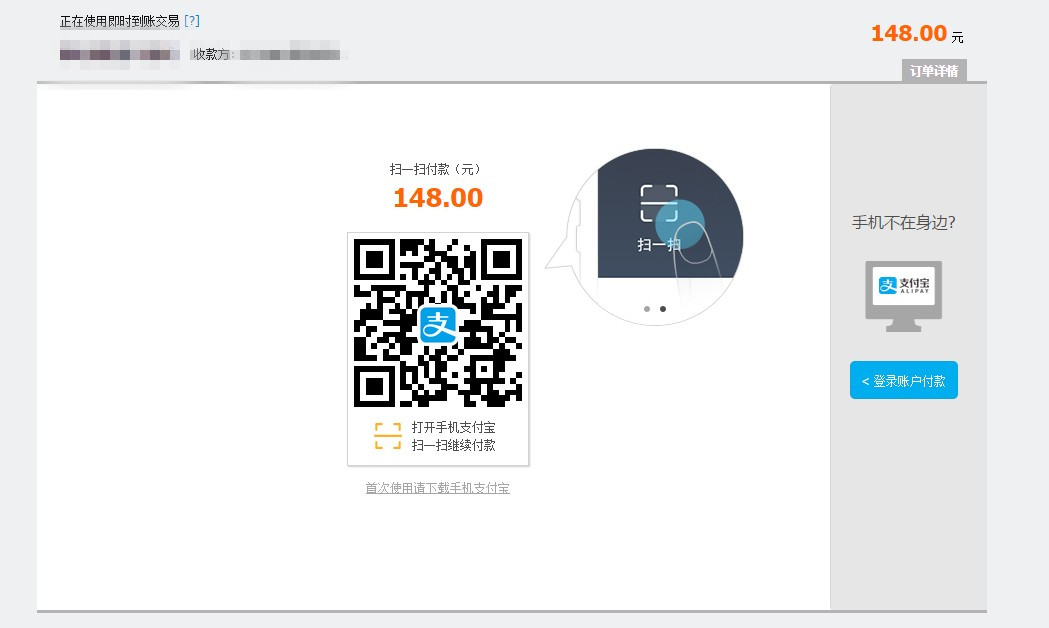
## 1 支付宝业务简介

买家在商户网站选择需购买的商品，填写订单信息后，点击立即购买。



网页跳转到支付宝收银台页面。

用户可以使用支付宝App扫一扫屏幕二维码，待手机提示付款后选择支付工具输入密码即可完成支付；



如果不使用手机支付，也可以点击上图右侧的“登录账户付款”，输入支付宝账号和支付密码登录PC收银台。

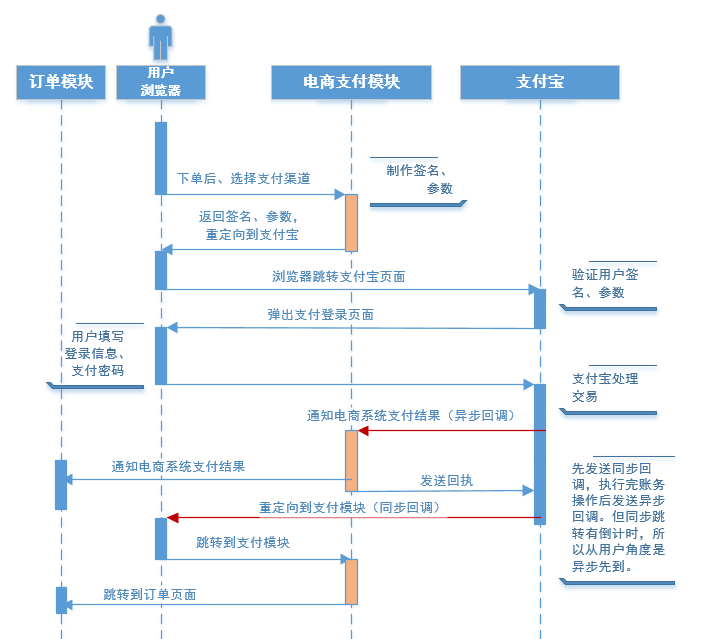


用户选择付款方式，输入支付密码后点击“确认付款”。

付款成功。



## 2 过程分析



## 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



## 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中：



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

点击查看



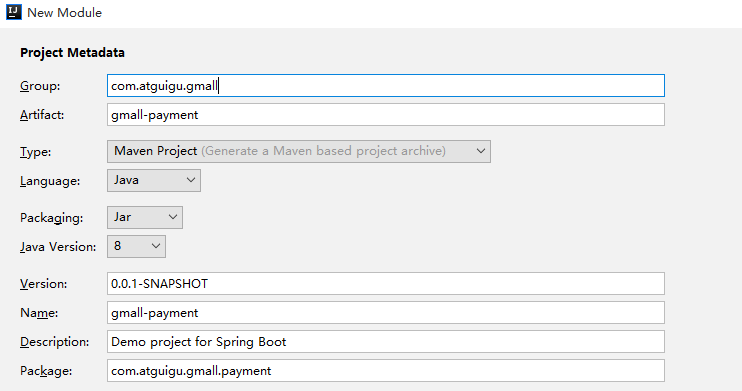
到此为止，电商网站可以访问支付宝的最基本的准备已经完成。

接下来搭建支付模块

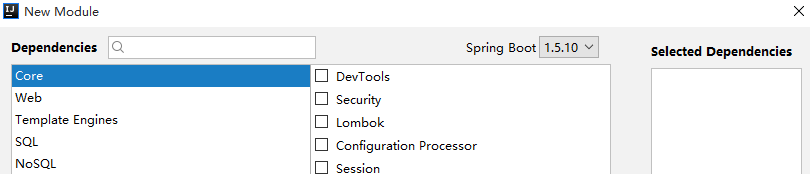
# 二、支付模块搭建

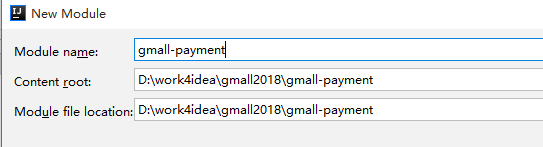
支付宝开发手册：<https://docs.open.alipay.com/270/105900/>

## 搭建gmall-payment项目【service-web】



这里由于支付的页面部分非常简单，便于讲解所以将支付模块的service 和web工程使用一个模块





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 http://maven.apache.org/xsd/maven-4.0.0.xsd">  <modelVersion>4.0.0</modelVersion>   <groupId>com.atguigu.gmall</groupId>  <artifactId>gmall-payment</artifactId>  <version>0.0.1-SNAPSHOT</version>  <packaging>jar</packaging>   <name>gmall-payment</name>  <description>Demo project for Spring Boot</description>   <parent>  <groupId>com.atguigu.gmall</groupId>  <artifactId>gmall-parent</artifactId>  <version>1.0-SNAPSHOT</version>  </parent>    <dependencies>   <dependency>  <groupId>com.atguigu.gmall</groupId>  <artifactId>gmall-interface</artifactId>  <version>1.0-SNAPSHOT</version>  </dependency>   <dependency>  <groupId>com.atguigu.gmall</groupId>  <artifactId>gmall-web-util</artifactId>  <version>1.0-SNAPSHOT</version>  </dependency>   <dependency>  <groupId>com.atguigu.gmall</groupId>  <artifactId>gmall-service-util</artifactId>  <version>1.0-SNAPSHOT</version>  </dependency>  </dependencies>   <build>  <plugins>  <plugin>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-maven-plugin</artifactId>  </plugin>  </plugins>  </build>   </project> |

application.properties

|  |
| --- |
| **server.port**=**8092  logging.level.root**=**debug  spring.thymeleaf.cache**=**false spring.thymeleaf.mode**=**LEGACYHTML5  spring.dubbo.application.name**=**payment spring.dubbo.registry.protocol**=**zookeeper spring.dubbo.registry.address**=**192.168.67.202:2181 spring.dubbo.base-package**=**com.atguigu.gmall spring.dubbo.protocol.name**=**dubbo spring.dubbo.consumer.timeout**=**1000000 spring.dubbo.consumer.check**=**false  spring.datasource.url**=**jdbc:mysql://localhost:3306/gmall?characterEncoding=UTF-8 spring.datasource.username**=**root spring.datasource.password**=**root  mapper.enum-as-simple-type**=**true mybatis.mapper-locations**=**classpath:mapper/\*Mapper.xml mybatis.configuration.mapUnderscoreToCamelCase**=**true** |

GmallPaymentApplication

|  |
| --- |
| @SpringBootApplication @ComponentScan(basePackages = **"com.atguigu.gmall"**) @MapperScan(basePackages = **"com.atguigu.gmall.payment.mapper"**) **public class** GmallPaymentApplication {   **public static void** main(String[] args) {  SpringApplication.*run*(GmallPaymentApplication.**class**, args);  } } |

## 导入sdk包

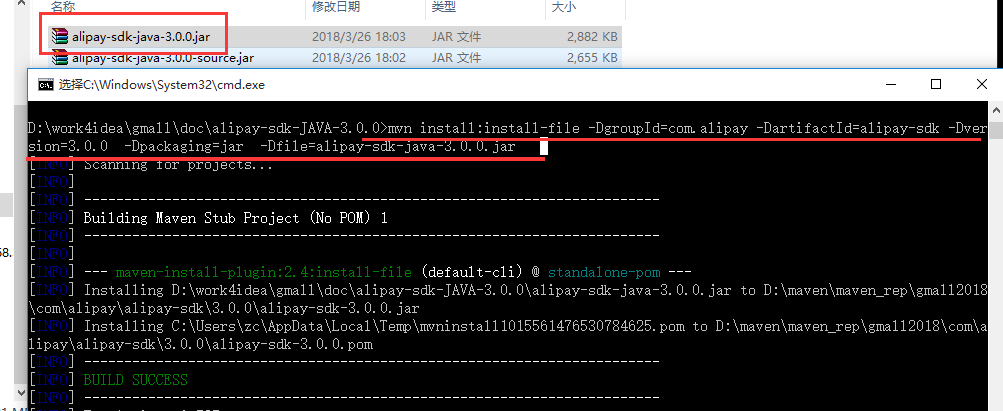
增加导入支付宝的sdk包到maven仓库。



调用支付宝的服务必须使用它的jar包，但是目前该jar包没有上传到网上的中心仓库上，所以只能手工导入到本地仓库中。

首先从蚂蚁金服的文档中心下载该jar包。

在jar包所在目录下，打开命令行工具 {cmd}

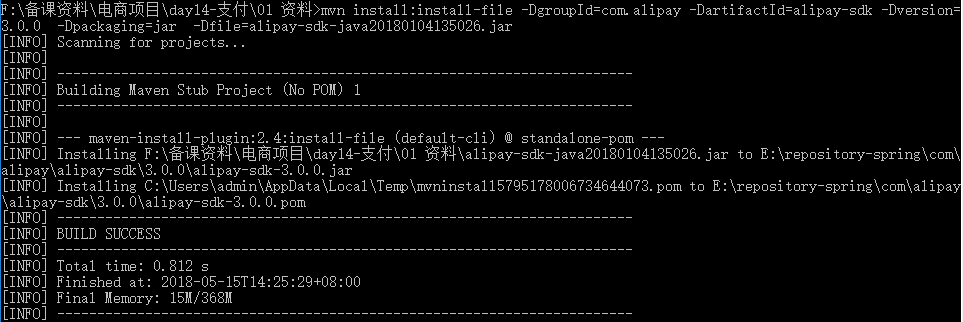


执行如下命令：

要想运行该命令，则必须配置maven的环境变量。

Mvn package:打包命令

|  |
| --- |
| mvn install:install-file -DgroupId=com.alipay -DartifactId=alipay-sdk -Dversion=3.0.0 -Dpackaging=jar -Dfile=alipay-sdk-java20180104135026.jar |



这样在pom.xml中就可以引入依赖

|  |
| --- |
| <**dependency**>  <**groupId**>com.alipay</**groupId**>  <**artifactId**>alipay-sdk</**artifactId**>  <**version**>3.0.0</**version**> </**dependency**> |

中心仓库

|  |
| --- |
| <!-- https://mvnrepository.com/artifact/com.alipay.sdk/alipay-sdk-java -->  <dependency>  <groupId>com.alipay.sdk</groupId>  <artifactId>alipay-sdk-java</artifactId>  <version>3.0.0</version>  </dependency> |

配置host

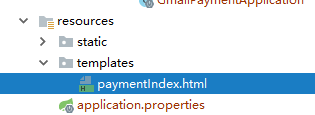


配置nginx

|  |
| --- |
| upstream payment.gmall.com{  server 192.168.67.1:8092;  }  server {  listen 80;  server\_name payment.gmall.com;  location / {  proxy\_pass http://payment.gmall.com;  proxy\_set\_header X-forwarded-for $proxy\_add\_x\_forwarded\_for;  }  } |

# 三、显示付款页面信息

引入静态文件及页面



PaymentController

|  |
| --- |
| @Controller **public class** PaymentController {   @Reference  OrderService **orderService**;  @RequestMapping(**"index"**)  @LoginRequire  **public** String index(HttpServletRequest request, Model model){  *// 获取订单的id* String orderId = request.getParameter(**"orderId"**);  OrderInfo orderInfo = **orderService**.getOrderInfo(orderId);  model.addAttribute(**"orderId"**,orderId);  model.addAttribute(**"totalAmount"**,orderInfo.getTotalAmount());  **return "index"**;  } } |

在 OrderServiceImpl中增加getOrderInfo方法

|  |
| --- |
| @Override **public** OrderInfo getOrderInfo(String orderId) {   OrderInfo orderInfo = **orderInfoMapper**.selectByPrimaryKey(orderId);  OrderDetail orderDetail = **new** OrderDetail();  orderDetail.setOrderId(orderId);  *// 将orderDetai 放入orderInfo 中* orderInfo.setOrderDetailList(**orderDetailMapper**.select(orderDetail));  **return** orderInfo;  } |

页面html

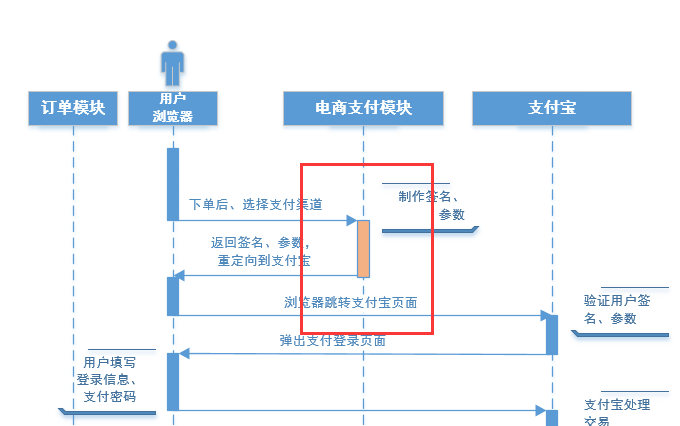
|  |
| --- |
| <**dd**>  <**span th:text="'订单提交成功，请尽快付款！订单号：'+${orderId}"**> </**span**>  <**span th:text="'应付金额'+${totalAmount}+'元'"**> </**span**> </**dd**> |

|  |
| --- |
| <**form method="post" id="paymentForm"**>  <**input type="hidden" name="orderId" th:value="${orderId}"** > </**form**> |

测试启动，支付渠道选择页面



# 四、跳转支付宝



## 1 分析

功能要求：

1. 制作支付宝需要的各种参数
2. 保存支付信息，作用：追踪交易状态、去重、对账
   1. 去重-保证接口的幂等性[去重。]
3. 帮助用户跳转到支付宝的页面

分析支付宝需要什么参数？

查看蚂蚁金服的文档中心中的电脑网站支付说明



这些参数可以一次性注入到阿里提供alipayClient中，以后就不用再赋值了。

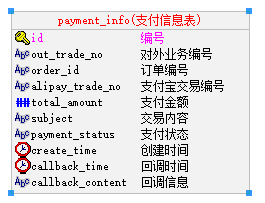
业务参数





## 2 支付信息的保存

表结构 payment\_info



|  |  |
| --- | --- |
| 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 |
| alipay\_trade\_no | 订单编号 初始为空，支付宝回调时生成 |
| 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 |

实体Bean

|  |
| --- |
| **public class** PaymentInfo **implements** Serializable{  @Column  @Id  @GeneratedValue(strategy = GenerationType.***IDENTITY***)  **private** String **id**;   @Column  **private** String **outTradeNo**;   @Column  **private** String **orderId**;   @Column  **private** String **alipayTradeNo**;   @Column  **private** BigDecimal **totalAmount**;   @Column  **private** String **Subject**;   @Column  **private** PaymentStatus **paymentStatus**;   @Column  **private** Date **createTime**;   @Column  **private** Date **callbackTime**;   @Column  **private** String **callbackContent**;  } |

PaymentStatus

|  |
| --- |
| **public enum** PaymentStatus {   ***UNPAID***(**"支付中"**),  ***PAID***(**"已支付"**),  ***PAY\_FAIL***(**"支付失败"**),  ***ClOSED***(**"已关闭"**);   **private** String **name** ;   PaymentStatus(String name) {  **this**.**name**=name;  } } |

## 3 PaymentController

分析 ：

1、通过orderId取得订单信息

2、组合对应的支付信息保存到数据库。

3、组合需要传给支付宝的参数。

4、根据返回的表单生成html，传给浏览器。。

|  |
| --- |
| @Controller **public class** PaymentController {   @Reference  **private** OrderService **orderService**;   @Autowired  **private** PaymentService **paymentService**;   @Autowired  AlipayClient **alipayClient**;   @RequestMapping(value = **"/alipay/submit"**,method = RequestMethod.***POST***)  @ResponseBody  **public** String submitPayment(HttpServletRequest request, HttpServletResponse response){  *// 获取订单Id* String orderId = request.getParameter(**"orderId"**);  *// 取得订单信息* OrderInfo orderInfo = **orderService**.getOrderInfo(orderId);  *// 保存支付信息* PaymentInfo paymentInfo = **new** PaymentInfo();  paymentInfo.setOrderId(orderId);  paymentInfo.setOutTradeNo(orderInfo.getOutTradeNo());  paymentInfo.setTotalAmount(orderInfo.getTotalAmount());  paymentInfo.setSubject(“-----”);  paymentInfo.setPaymentStatus(PaymentStatus.***UNPAID***);   *// 保存信息* **paymentService**.savePaymentInfo(paymentInfo);   *// 支付宝参数* AlipayTradePagePayRequest alipayRequest = **new** AlipayTradePagePayRequest();*//创建API对应的request* alipayRequest.setReturnUrl(AlipayConfig.*return\_payment\_url*);  alipayRequest.setNotifyUrl(AlipayConfig.*notify\_payment\_url*);*//在公共参数中设置回跳和通知地址   // 声明一个Map* Map<String,Object> bizContnetMap=**new** HashMap<>();  bizContnetMap.put(**"out\_trade\_no"**,paymentInfo.getOutTradeNo());  bizContnetMap.put(**"product\_code"**,**"FAST\_INSTANT\_TRADE\_PAY"**);  bizContnetMap.put(**"subject"**,paymentInfo.getSubject());  bizContnetMap.put(**"total\_amount"**,paymentInfo.getTotalAmount());  *// 将map变成json* String Json = JSON.*toJSONString*(bizContnetMap);  alipayRequest.setBizContent(Json);  String form=**""**;  **try** {  form = **alipayClient**.pageExecute(alipayRequest).getBody(); *//调用SDK生成表单* } **catch** (AlipayApiException e) {  e.printStackTrace();  }  response.setContentType(**"text/html;charset=UTF-8"**);  **return** form;  } |

OrderInfo实体类中添加内容： paymentInfo.setSubject();

|  |
| --- |
| **public** String getTradeBody(){  OrderDetail orderDetail = **orderDetailList**.get(0);  String tradeBody=orderDetail.getSkuName()+**"等"**+**orderDetailList**.size()+**"件商品"**;  **return** tradeBody; } |

添加config中的AlipayClient文件

## 4 PaymentService

接口：PaymentService

|  |
| --- |
| **public interface** PaymentService {  **void** savyPaymentInfo(PaymentInfo paymentInfo); } |

此处：将serviceImpl写在了同一个工程

|  |
| --- |
| @Service **public class** PaymentServiceImpl **implements** PaymentService{   @Autowired  PaymentInfoMapper **paymentInfoMapper**;   **public void** savPaymentInfo(PaymentInfo paymentInfo){  **paymentInfoMapper**.insertSelective(paymentInfo);  } } |

Mapper接口

PaymentInfoMapper

|  |
| --- |
| **public interface** PaymentInfoMapper **extends** Mapper<PaymentInfo>{ } |

OrderInfo 中添加 getTradeBody()

|  |
| --- |
| **public** String getTradeBody(){  OrderDetail orderDetail = **orderDetailList**.get(0);  String tradeBody=orderDetail.getSkuName()+**"等"**+**orderDetailList**.size()+**"件商品"**;  **return** tradeBody; } |

初始化AlipayClient

|  |
| --- |
| @Configuration @PropertySource("classpath:alipay.properties") 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 **public** AlipayClient alipayClient(){  AlipayClient alipayClient=**new** DefaultAlipayClient(**alipay\_url**,**app\_id**,**app\_private\_key**,***format***,***charset***, *alipay\_public\_key*,***sign\_type*** );  **return** alipayClient; }  } |

增加alipay.properties

在01 资料中拷贝即可！

问题：1 密钥如何得来，为什么有两个

利用工具生成 保存本地私钥 和支付宝公钥 把本地公钥上传给支付宝

2、url为什么有的用域名，有的用ip地址。

有域名的是本地返回地址！

Ip地址是发给阿里云的地址！公网上的！

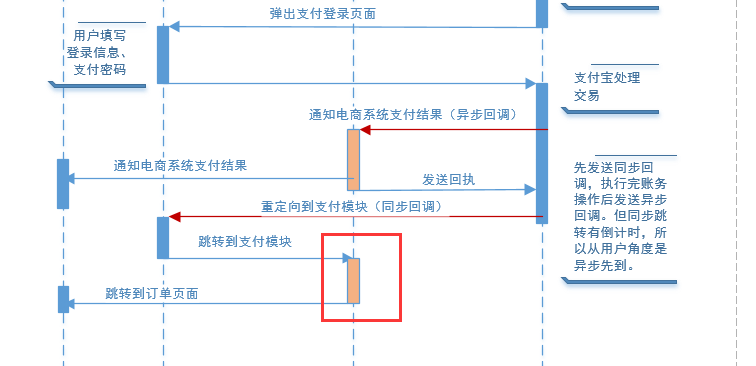
测试：



将金额修改小一点！

UPDATE order\_info SET total\_amount=0.5 , out\_trade\_no = 'ATGUIGU1526293773068470' WHERE id=5

# 五 支付后回调—同步回调



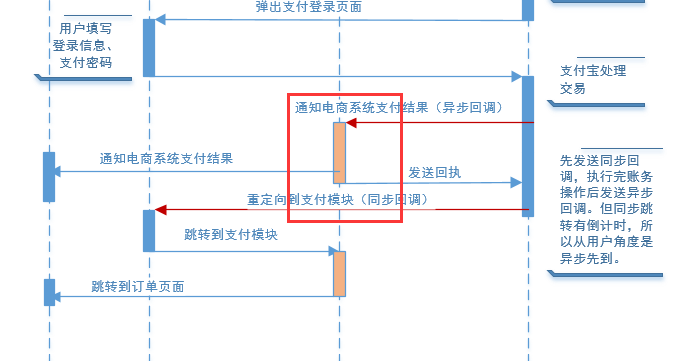
PaymentController

|  |
| --- |
| @RequestMapping(value = **"/alipay/callback/return"**,method = RequestMethod.***GET***) **public** String callbackReturn(){  **return "redirect:"**+AlipayConfig.*return\_order\_url*; } |

这里requestMapping对应的路径必须与之前传给支付宝的AlipayConfig.*return\_order\_url*保持一致。



# 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。

PaymentController

|  |
| --- |
| @RequestMapping(value = **"/alipay/callback/notify"**,method = RequestMethod.***POST***) @ResponseBody **public** String paymentNotify(@RequestParam Map<String,String> paramMap, HttpServletRequest request) **throws** AlipayApiException {  **boolean** flag = AlipaySignature.*rsaCheckV1*(paramMap, AlipayConfig.*alipay\_public\_key*, **"utf-8"**,AlipayConfig.***sign\_type***);  **if** (!flag){  **return "fail"**;  }  *// 判断结束* String trade\_status = paramMap.get(**"trade\_status"**);  **if** (**"TRADE\_SUCCESS"**.equals(trade\_status) || **"TRADE\_FINISHED"**.equals(trade\_status)){  *// 查单据是否处理* String out\_trade\_no = paramMap.get(**"out\_trade\_no"**);  PaymentInfo paymentInfo = **new** PaymentInfo();  paymentInfo.setOutTradeNo(out\_trade\_no);  PaymentInfo paymentInfoHas = **paymentService**.getPaymentInfo(paymentInfo);  **if** (paymentInfoHas.getPaymentStatus()==PaymentStatus.***PAID*** || paymentInfoHas.getPaymentStatus()==PaymentStatus.***ClOSED***){  **return "fail"**;  }**else** {  *// 修改* PaymentInfo paymentInfoUpd = **new** PaymentInfo();  *// 设置状态* paymentInfoUpd.setPaymentStatus(PaymentStatus.***PAID***);  *// 设置创建时间* paymentInfoUpd.setCallbackTime(**new** Date());  *// 设置内容* paymentInfoUpd.setCallbackContent(paramMap.toString());  **paymentService**.updatePaymentInfo(out\_trade\_no,paymentInfoUpd);  **return "success"**;  }  }  **return "fail"**; } |

PaymentServiceImpl

|  |
| --- |
| **public** PaymentInfo getpaymentInfo(PaymentInfo paymentInfo){  **return paymentInfoMapper**.selectOne(paymentInfo); }  @Override **public void** updatePaymentInfo(String outTradeNo, PaymentInfo paymentInfo) {  Example example = **new** Example(PaymentInfo.**class**);  example.createCriteria().andEqualTo(**"outTradeNo"**,outTradeNo);  **paymentInfoMapper**.updateByExampleSelective(paymentInfo,example); } |

测试：将程序的jar包放到阿里云服务器上

打jar包 install 放到linux一个目录下/root/app/gmall-payment-0.0.1-SNAPSHOT.jar

gmall-order-service-0.0.1-SNAPSHOT.jar

nohup java -jar gmall-payment-0.0.1-SNAPSHOT.jar > payment.out &

后台运行，并且将日志写在payment.out

tail -f 20 payment.out

查看20条数据

nohup java -jar gmall-order-service-0.0.1-SNAPSHOT.jar >order-service.out &

tail -f20 order-service.out

项目如何部署到阿里云服务器

项目打包到阿里云服务器上

Redis，zk，es，nginx，mysql, activeMq

配置文件中的mysql等连接地址应该阿里云的地址。

数据量非常大：mycat！

Docker！部署！镜像，容器。

Linux – 安装一个docker linux[Redis，zk，es，nginx，mysql, activeMq]

# 七、退款

直接在浏览器发起请求即可！

http://payment.gmall.com/refund?orderId=98

商户与客户协商一致的情况下，才可以退款！

|  |
| --- |
| @RequestMapping(**"refund"**) @ResponseBody **public** String refund(String orderId){  **boolean** flag = **paymentService**.refund(orderId);  System.***out***.println(**"flag:"**+flag);  **return** flag+**""**; } |
| **boolean** refund(String orderId); |
| **public boolean** refund(String orderId) {   *//AlipayClient alipayClient = new DefaultAlipayClient("https://openapi.alipay.com/gateway.do","app\_id","your private\_key","json","GBK","alipay\_public\_key","RSA2");* AlipayTradeRefundRequest request = **new** AlipayTradeRefundRequest();  PaymentInfo paymentInfo = getPaymentInfoByOrderId(orderId);   HashMap<String, Object> map = **new** HashMap<>();  map.put(**"out\_trade\_no"**,paymentInfo.getOutTradeNo());  map.put(**"refund\_amount"**, paymentInfoQuery.getTotalAmount());   request.setBizContent(JSON.*toJSONString*(map));  AlipayTradeRefundResponse response = **null**;  **try** {  response = **alipayClient**.execute(request);  } **catch** (AlipayApiException e) {  e.printStackTrace();  }  **if**(response.isSuccess()){  System.***out***.println(**"调用成功"**);  **return true**;  } **else** {  System.***out***.println(**"调用失败"**);  **return false**;  } } |

# 八、微信支付

## 微信扫码支付申请

微信扫码支付是商户系统按微信支付协议生成支付二维码，用户再用微信“扫一扫”完成支付的模式。该模式适用于PC网站支付、实体店单品或订单支付、媒体广告支付等场景。

申请步骤：（了解）

**第一步：注册公众号（类型须为：服务号）**

请根据营业执照类型选择以下主体注册：[个体工商户](http://kf.qq.com/faq/120911VrYVrA151009JB3i2Q.html)| [企业/公司](http://kf.qq.com/faq/120911VrYVrA151013MfYvYV.html)| [政府](http://kf.qq.com/faq/161220eaAJjE161220IJn6zU.html)| [媒体](http://kf.qq.com/faq/161220IFBJFv161220YnqAbQ.html)| [其他类型](http://kf.qq.com/faq/120911VrYVrA151013nYFZ7Z.html)。

**第二步：认证公众号**

公众号认证后才可申请微信支付，认证费：300元/年。

**第三步：提交资料申请微信支付**

登录公众平台，点击左侧菜单【微信支付】，开始填写资料等待审核，审核时间为1-5个工作日内。

**第四步：开户成功，登录商户平台进行验证**

资料审核通过后，请登录联系人邮箱查收商户号和密码，并登录商户平台填写财付通备付金打的小额资金数额，完成账户验证。

**第五步：在线签署协议**

本协议为线上电子协议，签署后方可进行交易及资金结算，签署完立即生效。

## 开发文档

微信支付接口调用的整体思路：

按API要求组装参数，以XML方式发送（POST）给微信支付接口（URL）,微信支付接口也是以XML方式给予响应。程序根据返回的结果（其中包括支付URL）生成二维码或判断订单状态。

在线微信支付开发文档：

<https://pay.weixin.qq.com/wiki/doc/api/index.html>

1. appid：微信公众账号或开放平台APP的唯一标识
2. mch\_id：商户号 (配置文件中的partner)
3. partnerkey：商户密钥
4. sign:数字签名, 根据微信官方提供的密钥和一套算法生成的一个加密信息, 就是为了保证交易的安全性

## 微信支付SDK



添加依赖

|  |
| --- |
| <dependency>  <groupId>com.github.wxpay</groupId>  <artifactId>wxpay-sdk</artifactId>  <version>0.0.3</version>  </dependency> |

我们主要会用到微信支付SDK的以下功能：

1. 获取随机字符串

|  |
| --- |
| WXPayUtil.*generateNonceStr*() |

1. MAP转换为XML字符串（自动添加签名）,两步并一步 （转xml和签名 ）

|  |
| --- |
| WXPayUtil.*generateSignedXml*(param, partnerkey) |

1. XML字符串转换为MAP

|  |
| --- |
| WXPayUtil.*xmlToMap*(result) |

## 功能开发：

场景：用户扫描商户展示在各种场景的二维码进行支付

使用案例：

线下：家乐福超市、7-11便利店、上品折扣线下店等

线上：大众点评网站、携程网站、唯品会、美丽说网站等

开发模式：

模式一：商户在后台给你生成二维码，用户打开扫一扫

模式二：商户后台系统调用微信支付【[统一下单API](https://pay.weixin.qq.com/wiki/doc/api/native.php?chapter=9_1)】生成预付交易，将接口返回的链接生成二维码，用户扫码后输入密码完成支付交易。注意：**该模式的预付单有效期为2小时，**过期后无法支付。

微信支付：生成xml 发送请求

导入umd 到static 目录下，将工具类添加到gmall-common-util中

在application.properties 中添加商户信息

|  |
| --- |
| **appid**=**wxf913bfa3a2c7eeeb partner**=**1543338551 partnerkey**=**atguigu3b0kn9g5v426MKfHQH7X8rKwb** |

编写控制器

|  |
| --- |
| @PostMapping(**"/wx/submit"**) **public** Map wxSubmit(String orderId) **throws** Exception {   OrderInfo orderInfo = **orderService**.getOrderInfo(orderId);   Map paramMap=**new** HashMap();  paramMap.put(**"appid"**,**appid**);  paramMap.put(**"mch\_id"**,**partner**);  paramMap.put(**"nonce\_str"**,WXPayUtil.*generateNonceStr*());  paramMap.put(**"body"**,orderInfo.genSubject());  paramMap.put(**"out\_trade\_no"**, **"ATGUIGU-"**+orderId+**"-"**+System.*currentTimeMillis*());  paramMap.put(**"total\_fee"**,orderInfo.getTotalAmount().multiply(**new** BigDecimal(100)).toBigInteger().toString());  paramMap.put(**"spbill\_create\_ip"**,**"127.0.0.1"**);  paramMap.put(**"notify\_url"**,**"http://269026wg43.zicp.vip/wx/callback/notify"**);  paramMap.put(**"trade\_type"**,**"NATIVE"**);   String xmlParam=WXPayUtil.*generateSignedXml*(paramMap,**partnerkey**);    HttpClient httpClient = **new** HttpClient(**"https://api.mch.weixin.qq.com/pay/unifiedorder"**);   httpClient.setXmlParam(xmlParam);   httpClient.post();   String content = httpClient.getContent(); *//得到返回结果的xml* Map<String, String> resultMap = WXPayUtil.*xmlToMap*(content);   **if**(resultMap.get(**"code\_url"**)!=**null**){  String code\_url = resultMap.get(**"code\_url"**);  *// HashMap<Object, Object> hashMap = new HashMap<>();* **return** resultMap;  }**else**{  System.***out***.println(resultMap.get(**"return\_code"**));  System.***out***.println(resultMap.get(**"return\_msg"**));  **return null**;  }   } |
| **public class** PaymentServiceImpl **implements** PaymentService {  *// 服务号Id* @Value(**"${appid}"**) **private** String **appid**; *// 商户号Id* @Value(**"${partner}"**) **private** String **partner**; *// 密钥* @Value(**"${partnerkey}"**) **private** String **partnerkey**;  @Override **public** Map createNative(String orderId, String total\_fee) {  *//1.创建参数* Map<String,String> param=**new** HashMap();*//创建参数* param.put(**"appid"**, **appid**);*//公众号* param.put(**"mch\_id"**, **partner**);*//商户号* param.put(**"nonce\_str"**, WXPayUtil.*generateNonceStr*());*//随机字符串* param.put(**"body"**, **"尚硅谷"**);*//商品描述* param.put(**"out\_trade\_no"**, orderId);*//商户订单号* param.put(**"total\_fee"**,total\_fee);*//总金额（分）* param.put(**"spbill\_create\_ip"**, **"127.0.0.1"**);*//IP* param.put(**"notify\_url"**, **"http://order.gmall.com/trade"**);*//回调地址(随便写)* param.put(**"trade\_type"**, **"NATIVE"**);*//交易类型* **try** {  *//2.生成要发送的xml* String xmlParam = WXPayUtil.*generateSignedXml*(param, **partnerkey**);  System.***out***.println(xmlParam);  HttpClient client=**new** HttpClient(**"https://api.mch.weixin.qq.com/pay/unifiedorder"**);  client.setHttps(**true**);  client.setXmlParam(xmlParam);  client.post();  *//3.获得结果* String result = client.getContent();  System.***out***.println(result);  Map<String, String> resultMap = WXPayUtil.*xmlToMap*(result);  Map<String, String> map=**new** HashMap<>();  map.put(**"code\_url"**, resultMap.get(**"code\_url"**));*//支付地址* map.put(**"total\_fee"**, total\_fee);*//总金额* map.put(**"out\_trade\_no"**,orderId);*//订单号* **return** map;  } **catch** (Exception e) {  e.printStackTrace();  **return new** HashMap<>();  } }  } |
| @PostMapping("/wx/callback/notify") public String notify(HttpServletRequest request, HttpServletResponse response ) throws Exception {  *// 0 获得值* ServletInputStream inputStream = request.getInputStream();  String xmlString = StreamUtil.*inputStream2String*(inputStream,"utf-8");   *// 1 验签* if( WXPayUtil.*isSignatureValid*(xmlString,partnerkey )){  *//2 判断状态* Map<String, String> paramMap = WXPayUtil.*xmlToMap*(xmlString);  String result\_code = paramMap.get("result\_code");  if(result\_code!=null&&result\_code.equals("SUCCESS")){  *// 3 更新支付状态 包发送 消息给订单   // 4 准备返回值 xml* HashMap<String, String> returnMap = new HashMap<>();  returnMap.put("return\_code","SUCCESS");  returnMap.put("return\_msg","OK");   String returnXml = WXPayUtil.*mapToXml*(returnMap);  response.setContentType("text/xml");  System.*out*.println("交易编号："+paramMap.get("out\_trade\_no")+"支付成功！");  return returnXml;   }else{  System.*out*.println(paramMap.get("return\_code")+"---"+paramMap.get("return\_msg"));  }  }  return null; } |

问题：

微信支付？支付宝支付？

如果用了微信支付，如何防止用户再次使用支付宝支付？

解决方案：

1. 支付日志记录每个订单支付的状态
   1. 无论微信支付，支付宝支付，都会有同步，异步回调
   2. 重点判断异步回调{给商家看，根据异步回调结果修改日志记录中的支付状态。}
   3. 在每种支付方式的控制器中，要做一个判断{订单的支付状态如果订单支付状态为已支付，则不生成二维码，给一个提示即可！}
   4. 如果支付状态只保存在数据库中，则会在高并发情况下会对数据库产生压力。 优化方案：将支付的状态放入缓存中！
      1. Set(key,value) key=user:orderId:info value=PaymentStatus.PAID
      2. 状态 如果是PaymentStatus.PAID 或者 PaymentStatus.CLOSE 都不能生成二维码
      3. 什么时候可以将支付状态删除：减库存完成，客户确认收获以后。