谷粒商城

版本：V 1.0

支付宝接口



# 一、支付业务

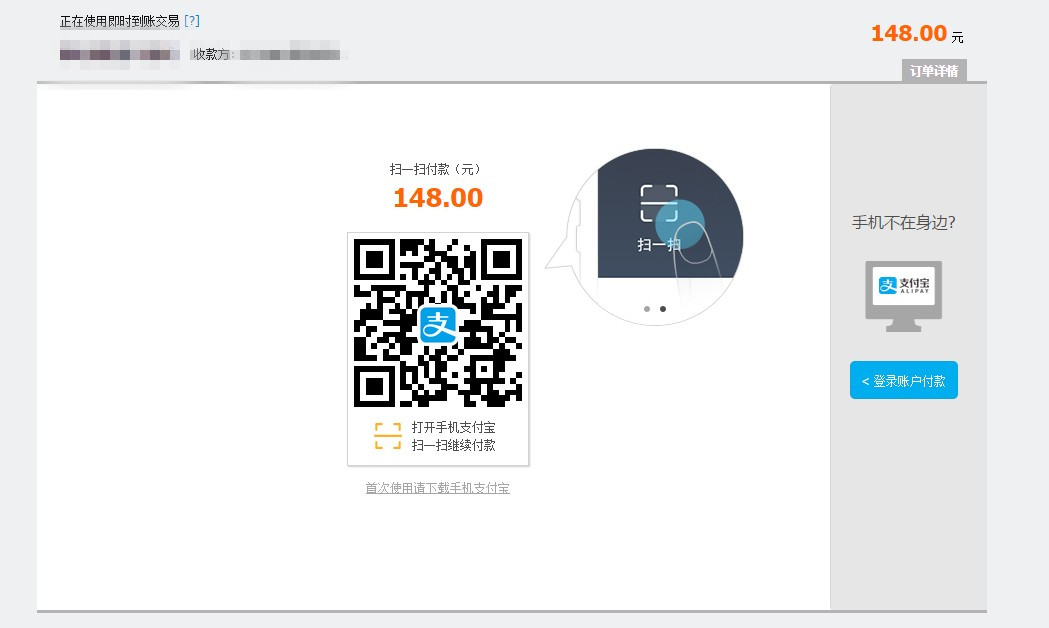
## 1 支付宝业务简介

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。



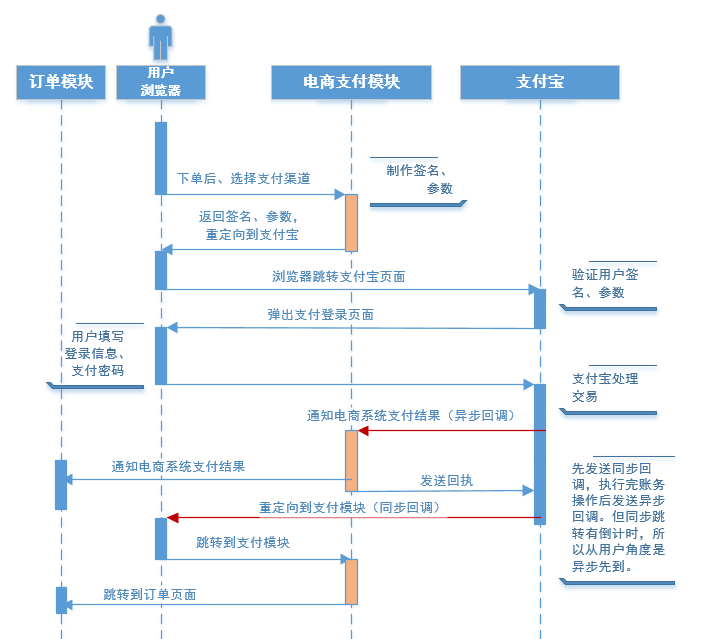
用户选择付款方式，输入支付密码后点击“确认付款”。



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/>

## 5 其他api

https://docs.open.alipay.com/62/104743



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。

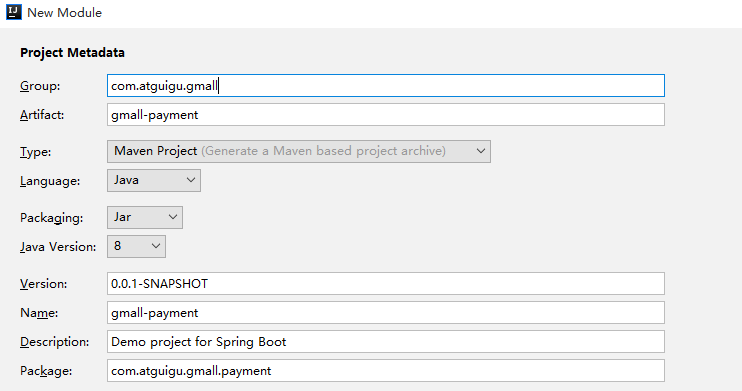
点击查看



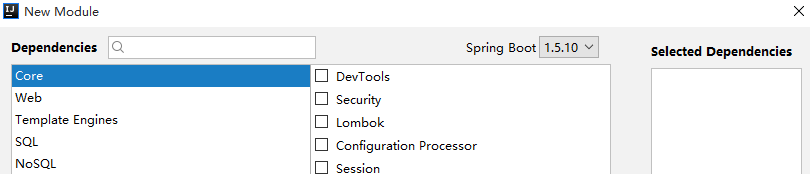
到此为止，电商网站可以访问支付宝的最基本的准备已经完成。

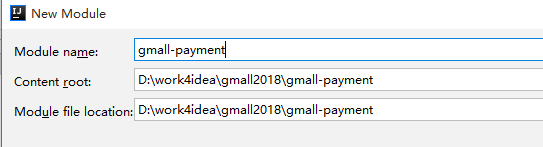
接下来搭建支付模块

# 二、支付模块搭建



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=8090  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=192.168.67.163:2181 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://59.110.141.236:3306/gmall?characterEncoding=UTF-8 spring.datasource.username=root spring.datasource.password=123456   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);  } } |

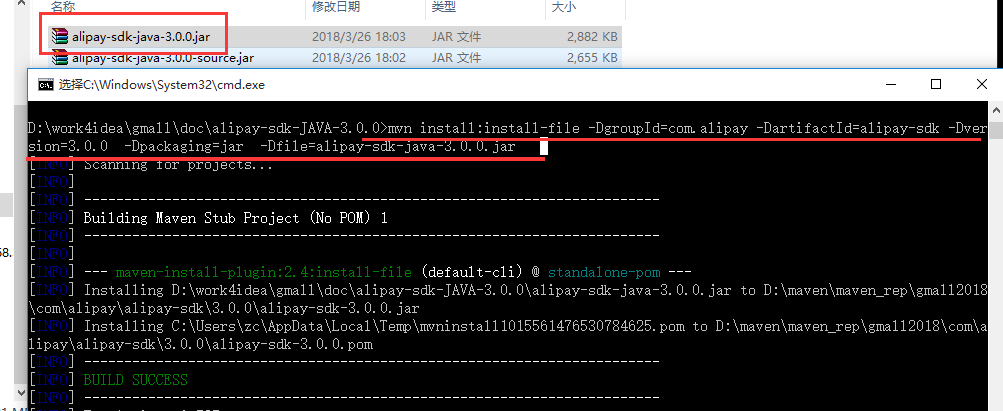
增加导入支付宝的sdk包到maven仓库。



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



执行如下命令

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

配置host

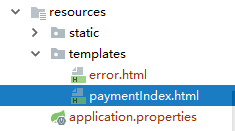


配置nginx

|  |
| --- |
| upstream payment.gmall.com{  server 192.168.67.1:8090;  }  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、支付渠道的选择页面

引入静态文件及页面



PaymentController

|  |
| --- |
| @Controller **public class** PaymentController {   @Reference  OrderService **orderService**;   @RequestMapping(value = **"index"**,method = RequestMethod.***GET***)  **public** String paymentIndex(@RequestParam(**"orderId"**)String orderId, Model model){    OrderInfo orderInfo=**orderService**.getOrderInfo( orderId );    model.addAttribute(**"orderId"**,orderInfo.getId());  model.addAttribute(**"totalAmount"**,orderInfo.getTotalAmount());    **return "paymentIndex"**;  } } |

在 OrderServiceImpl中增加getOrderInfo方法

|  |
| --- |
| @Override **public** OrderInfo getOrderInfo(String id) {  OrderInfo orderInfo = **orderInfoMapper**.selectByPrimaryKey(id);  OrderDetail orderDetailQuery=**new** OrderDetail();  orderDetailQuery.setOrderId(orderInfo.getId());  List<OrderDetail> orderDetailList = **orderDetailMapper**.select(orderDetailQuery);  orderInfo.setOrderDetailList(orderDetailList);   **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**> |

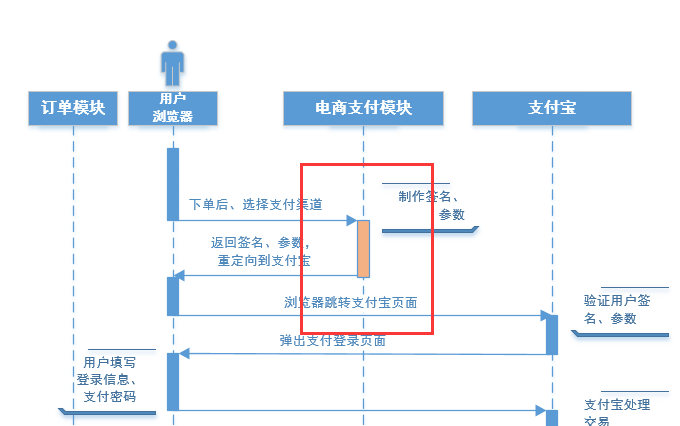
js

|  |
| --- |
| <**script type="text/javascript"**>  *$*(**function**() {  *$*(**"#paymentButton"**).click(**function** () {  *$*(**"#paymentForm"**).attr(**"action"**,**"/"**+*$*(**"input[type='radio']:checked"**).val()+**"/submit"**) ;  *$*(**"#paymentForm"**).submit();  **console**.log(*$*(**"#paymentForm"**).html()) ;  **console**.log(*$*(**"#paymentForm"**).attr(**"action"**)) ;  })    }) |

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
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。

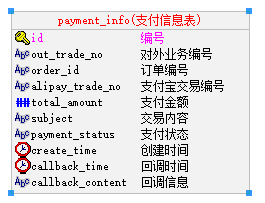
业务参数





## 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 |

## 3 类AlipayConfig整合到项目

|  |
| --- |
| import com.alipay.api.AlipayClient;  import com.alipay.api.DefaultAlipayClient;  import org.springframework.beans.factory.annotation.Value;  import org.springframework.context.annotation.Bean;  import org.springframework.context.annotation.Configuration;  import org.springframework.context.annotation.PropertySource;  /\*\*  \* @param  \* @return  \*/  @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;  } |

## 4 PaymentServiceImpl

|  |
| --- |
| @Override **public void** savePaymentInfo(PaymentInfo paymentInfo) {  *//必须保证每个订单只有唯一的支付信息，所以如果之前已经有了该笔订单的支付信息，那么只更新时间* PaymentInfo paymentInfoQuery=**new** PaymentInfo();  paymentInfoQuery.setOrderId(paymentInfo.getOrderId());   PaymentInfo paymentInfoExists = **paymentInfoMapper**.selectOne(paymentInfoQuery);  **if**(paymentInfoExists!=**null**){  paymentInfoExists.setCreateTime(**new** Date());  **paymentInfoMapper**.updateByPrimaryKey(paymentInfoExists);  **return**;  }   paymentInfo.setCreateTime(**new** Date());  **paymentInfoMapper**.insertSelective(paymentInfo); } |

PaymentInfoMapper

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

OrderInfo

|  |
| --- |
| //生成摘要  public String getOrderSubject(){  String body="";  if(orderDetailList!=null&&orderDetailList.size()>0){  body= orderDetailList.get(0).getSkuName();  }  body+="等"+getTotalSkuNum()+"件商品";  return body;  }  public Integer getTotalSkuNum(){  Integer totalNum=0;  for (OrderDetail orderDetail : orderDetailList) {  totalNum+= orderDetail.getSkuNum();  }  return totalNum; } |

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;  } } |

## 5 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html，传给浏览器。

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

|  |
| --- |
| @RequestMapping(value = **"/alipay/submit"**,method = RequestMethod.***POST***) @ResponseBody **public** ResponseEntity<String> paymentAlipay(HttpServletRequest request, HttpServletResponse httpServletResponse, Model model) **throws** IOException {    String orderId = request.getParameter(**"orderId"**);  **if**(orderId==**null**||orderId.length()==0){  **return** ResponseEntity.*status*(HttpStatus.***BAD\_REQUEST***).build();  }  *//获取订单信息* OrderInfo orderInfo= **orderService**.getOrderInfo( orderId );  **if**(orderInfo==**null** ){  *//没有对应的订单* **return** ResponseEntity.*status*(HttpStatus.***BAD\_REQUEST***).build();  }  *//保存支付信息* PaymentInfo paymentInfo =**new** PaymentInfo();  paymentInfo.setOrderId(orderId);  paymentInfo.setOutTradeNo(orderInfo.getOutTradeNo());  paymentInfo.setSubject(orderInfo.getOrderSubject());  paymentInfo.setPaymentStatus(PaymentStatus.***UNPAID***);  paymentInfo.setTotalAmount(orderInfo.getTotalAmount());  **paymentService**.savePaymentInfo(paymentInfo);    *//利用支付宝客户端生成表单页面* AlipayTradePagePayRequest alipayRequest=**new** AlipayTradePagePayRequest();   alipayRequest.setReturnUrl(AlipayConfig.*return\_payment\_url*);  alipayRequest.setNotifyUrl(AlipayConfig.*notify\_payment\_url*);   Map<String,String> paramMap=**new** HashMap<>();  paramMap.put(**"out\_trade\_no"**,paymentInfo.getOutTradeNo());  paramMap.put(**"product\_code"**,**"FAST\_INSTANT\_TRADE\_PAY"**);  paramMap.put(**"total\_amount"**,paymentInfo.getTotalAmount().toString());  paramMap.put(**"subject"**,paymentInfo.getSubject());  String paramJson = JSON.*toJSONString*(paramMap);  alipayRequest.setBizContent(paramJson);  String form=**""**;  **try** {  form = **alipayClient**.pageExecute(alipayRequest).getBody();  } **catch** (AlipayApiException e) {  e.printStackTrace();  }   httpServletResponse.setContentType(**"text/html;charset=utf-8"** );   *//把表单html打印到客户端浏览器* **return** ResponseEntity.*ok*().body(form) ; } |

测试页面



## 5 配置alipay.properties

增加alipay.properties

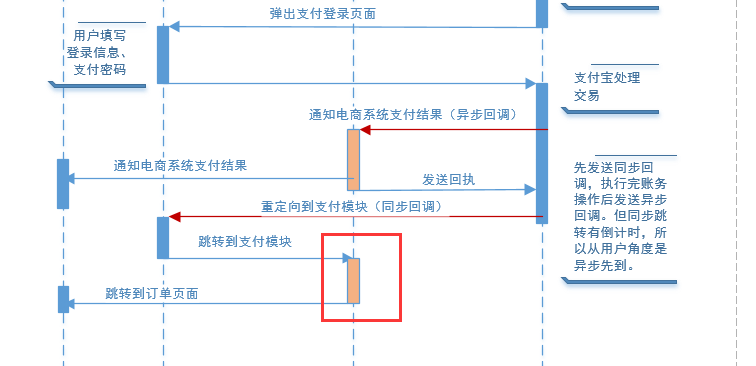
|  |
| --- |
| **alipay\_url**=**https://openapi.alipay.com/gateway.do  app\_id**=**2018020102122556  app\_private\_key**=**  alipay\_public\_key**=**MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAhkZi6W0wn/prX+NIIF9ATb5Z8ReKK4hFYtBrweDfGHD1mNW7YIZY4G5hE7S2Sry8eFXlFgSlBWlJ4fVnDaK9MkVThpwE2H65ooVlK/wLuyPqovIVpMt/utva5Ayuzv7eQOWK45FdLDNDlK8QLoBko6SS+YbnWnf7a+mrf4NAS4UFClpfe8Byqe8XIraO2Cg4Ko5Y5schX39rOAH8GlLdgqQRYVQ2dCnkIQ+L+I4Cy9Mvw3rIkTwt3MBU+AqREXY4r5Bn6cmmX/9MAJbFqrofGiUAqG+qbjTcZAzgNPfuiD0zXgt/YYjMQMzck75BOmwnYOam2ajODUSQn8Xybsa7wQIDAQAB** *# 同步回调地址 重定向地址本地浏览器* **return\_payment\_url**=**http://payment.gmall.com:8090/alipay/callback/return** *# 异步通知地址 公网接口(webService)* **notify\_payment\_url**=**http://60.205.215.91/alipay/callback/notify return\_order\_url**=**http://order.gmall.com:8086/orderList** |

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。

# 五、支付后回调—同步回调

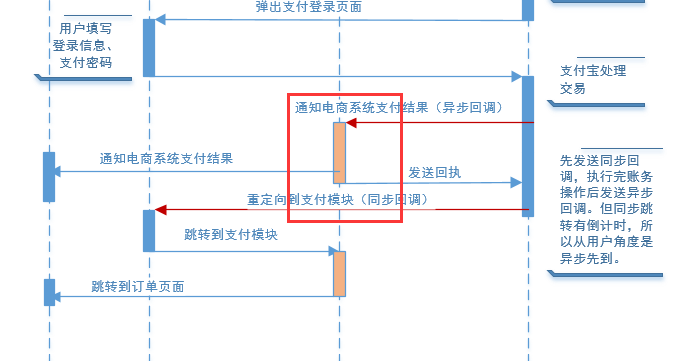


PaymentController

|  |
| --- |
| @RequestMapping(value=**"/alipay/callback/return"**,method = RequestMethod.***GET***) **public** String callbackReturn(HttpServletRequest request, Model model) **throws** UnsupportedEncodingException {  System.***out***.println(**" callback return to "**+AlipayConfig.*return\_order\_url*);   **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alipayRequest.setReturnUrl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 callbackNotify(@RequestParam Map<String,String> paramMap) {  System.***out***.println(**" ----------callbackstart 支付宝开始回调"**+paramMap.toString() );  *//验证签名* **boolean** isCheckPass=**false**;  **try** {  isCheckPass = AlipaySignature.*rsaCheckV1*(paramMap, AlipayConfig.*alipay\_public\_key*, AlipayConfig.***charset***, AlipayConfig.***sign\_type***);  } **catch** (AlipayApiException e) {  e.printStackTrace();  }  **if**(!isCheckPass){  System.***out***.println(**" ----------验签不通过！！"** );  **return "验签不通过！！"**;  }  System.***out***.println(**" ----------验签通过！！"** );  *//验证成功标志* String trade\_status = paramMap.get(**"trade\_status"**);  **if**(**"TRADE\_SUCCESS"**.equals(trade\_status)){  *//检查当前支付状态* String outTradeNo = paramMap.get(**"out\_trade\_no"**);  PaymentInfo paymentInfoQuery=**new** PaymentInfo();  paymentInfoQuery.setOutTradeNo(outTradeNo);  PaymentInfo paymentInfo = **paymentService**.getPaymentInfo(paymentInfoQuery);  **if** (paymentInfo==**null**) {  **return "error: not exists out\_trade\_no:"**+outTradeNo;  }  System.***out***.println(**"检查是否已处理= "** +outTradeNo );  **if**(paymentInfo.getPaymentStatus()==PaymentStatus.***PAID***){  *//如果已经处理过了 就直接返回成功标志* System.***out***.println(**" 已处理= "** +outTradeNo );  **return "success"**;  }**else** {  *//先更新支付状态* System.***out***.println(**" 未处理，更新状态= "** +outTradeNo );  PaymentInfo paymentInfo4Upt=**new** PaymentInfo();  paymentInfo4Upt.setPaymentStatus(PaymentStatus.***PAID***);  paymentInfo4Upt.setConfirmTime(**new** Date());  paymentInfo.setCallbackContent(paramMap.toString());  **paymentService**.updatePaymentInfoByOutTradeNo(outTradeNo,paymentInfo4Upt);   *//发送通知给订单* **paymentService**.sendPaymentResult2MQ(paymentInfo.getOrderId());  **return "success"**;   }   }  **return ""**;  } |

PaymentServiceImpl

|  |
| --- |
| */\*\*\*  \* 根据outTradeNo 更新支付信息  \** ***@param outTradeNo*** *\** ***@param paymentInfo*** *\*/* **public void** updatePaymentInfoByOutTradeNo(String outTradeNo , PaymentInfo paymentInfo){  Example example=**new** Example(PaymentInfo.**class**);  example.createCriteria().andEqualTo(**"outTradeNo"**,outTradeNo);   **paymentInfoMapper**.updateByExampleSelective(paymentInfo,example);  }   */\*\*  \* 根据outTrade查询支付信息  \** ***@param outTradeNo*** *\** ***@return*** *\*/* **public** PaymentInfo getPaymentInfoByOutTradeNo(String outTradeNo) {  Example example=**new** Example(PaymentInfo.**class**);  example.createCriteria().andEqualTo(**"outTradeNo"**,outTradeNo);   PaymentInfo paymentInfo = **paymentInfoMapper**.selectOneByExample(example);  **return** paymentInfo; } |

测试将应用程序发布到aliyun服务器上，支付并观察支付宝的回调。

