

支付宝接口

版本：V 1.0

www.atguigu.com

一、支付业务

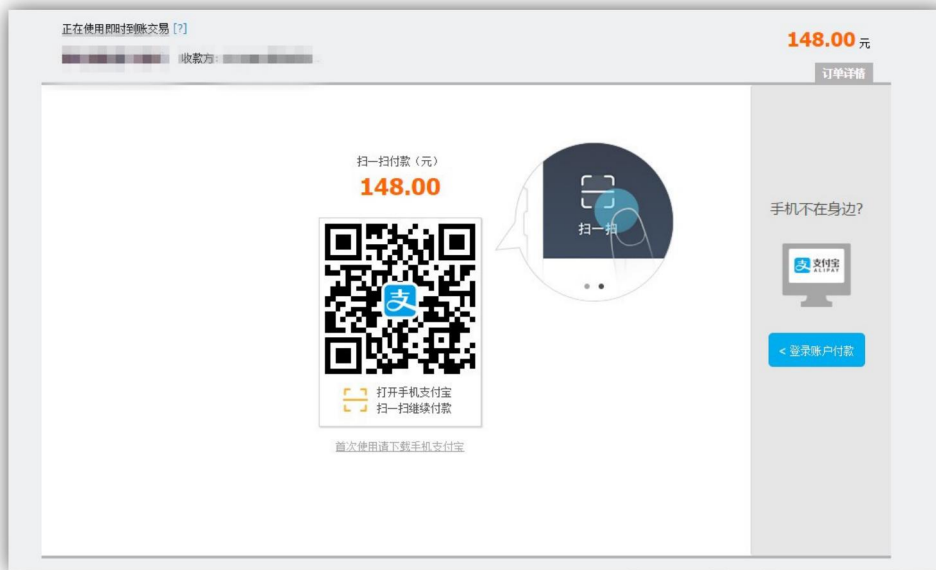
1 支付宝业务简介

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。



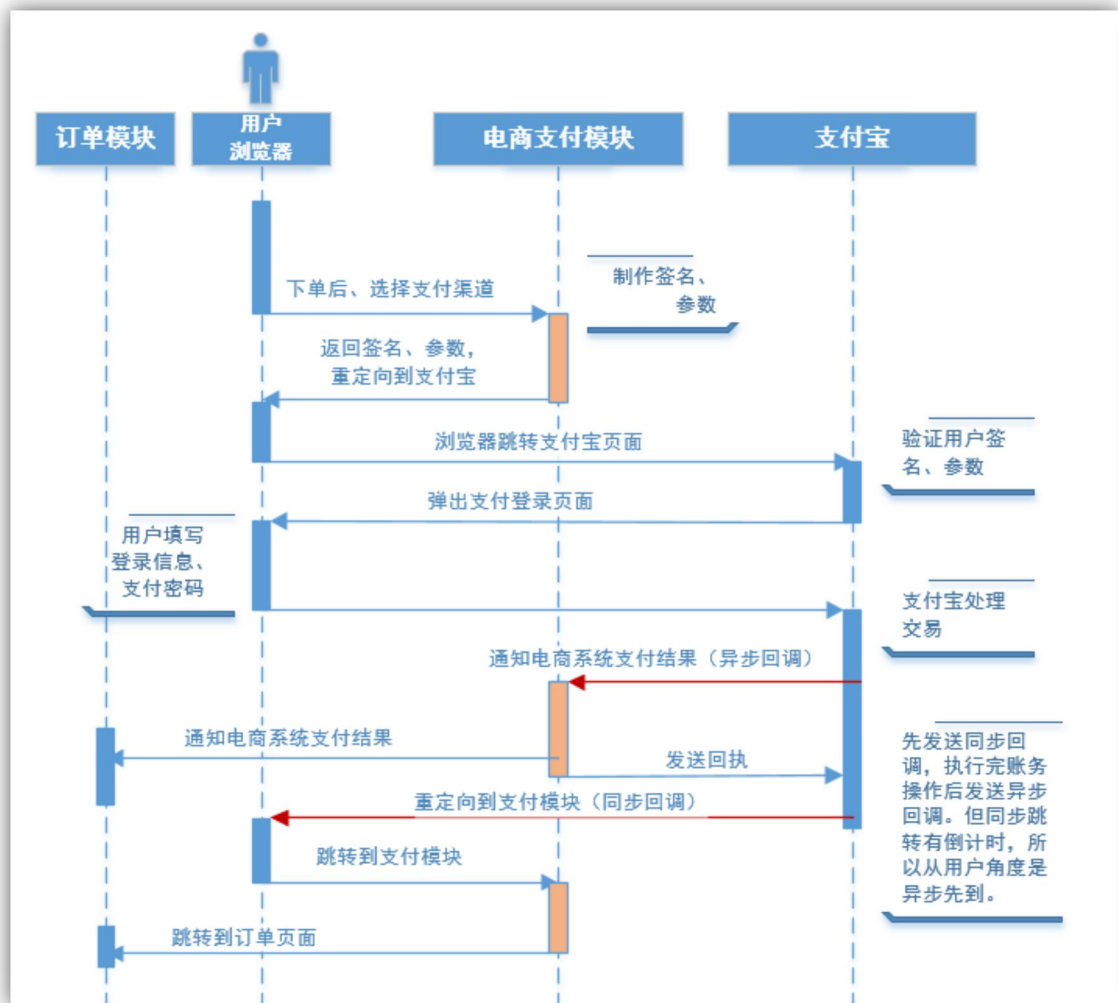
用户选择付款方式，输入支付密码后点击“确认付款”。



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

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)

4 申请步骤:

- 1、支付宝商家中心中申请 <https://www.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

概览



应用2.0签约2018020102358605

自用型应用

推广服务



发布到服务市场

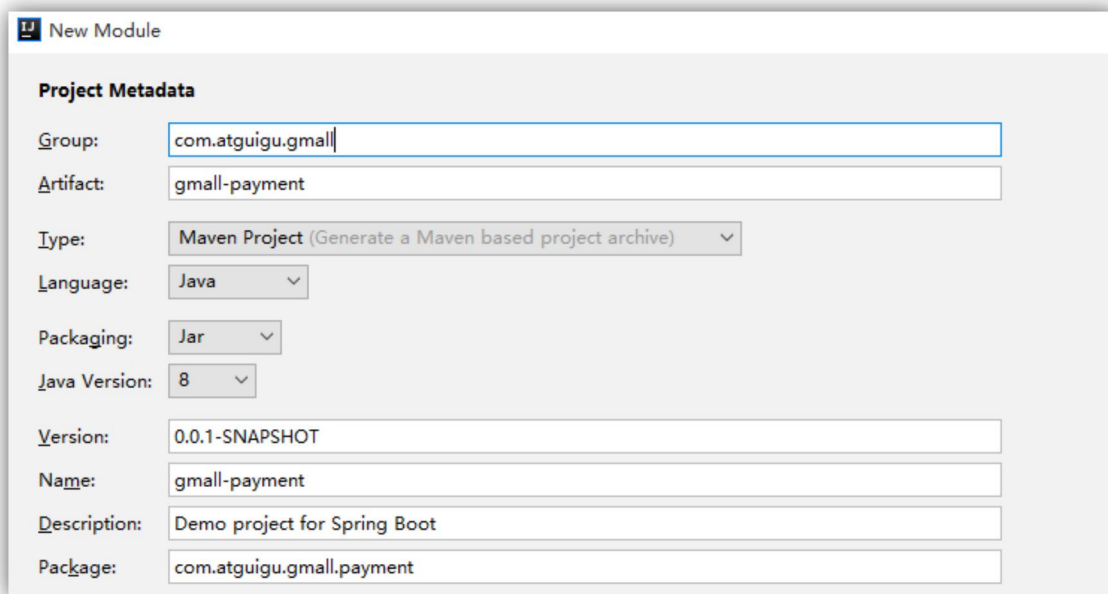
功能选项 已经添加1项个人使用功能 + 继续添加 查看更多能力

功能名称	功能介绍	是否需要签约	状态	操作
电脑网站支付	用户通过支付宝PC网站收银台完成支付，交易款项即时给到商户支付宝账户。	需签约	已生效	删除

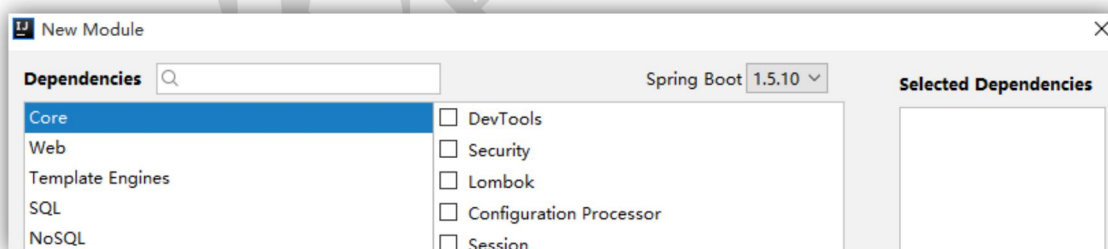
到此为止，电商网站可以访问支付宝的最基本的准备已经完成。

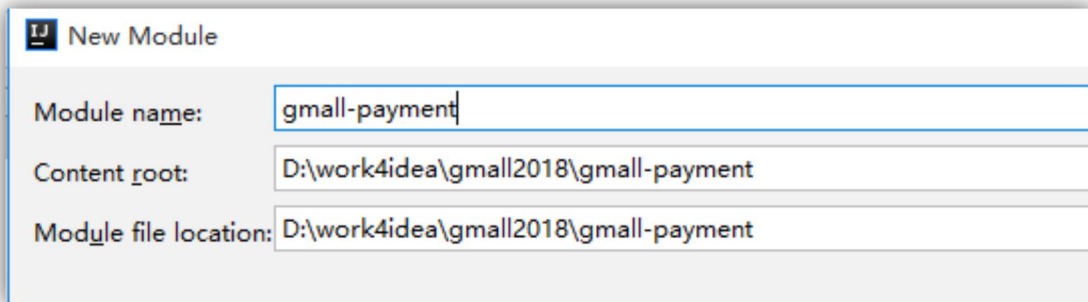
接下来搭建支付模块

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





The image shows a 'New Module' dialog box from an IDE. It contains three input fields: 'Module name' with the value 'gmall-payment', 'Content root' with the value 'D:\work4idea\gmall2018\gmall-payment', and 'Module file location' with the value 'D:\work4idea\gmall2018\gmall-payment'.

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>
  </dependencies>
</project>
```

```
<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);
    }
}
```

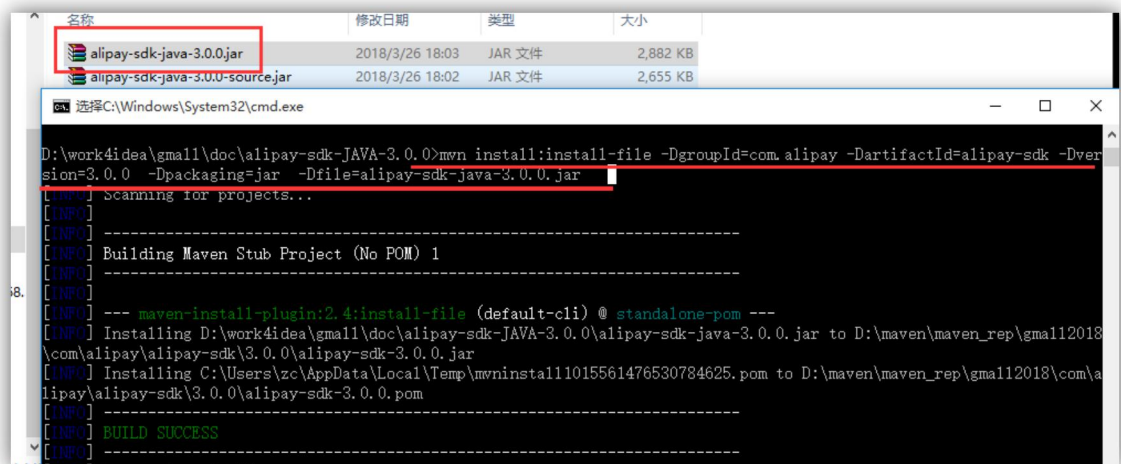
增加导入支付宝的 sdk 包到 maven 仓库。



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

```
# common
0.0.0.0 account.jetbrains.com

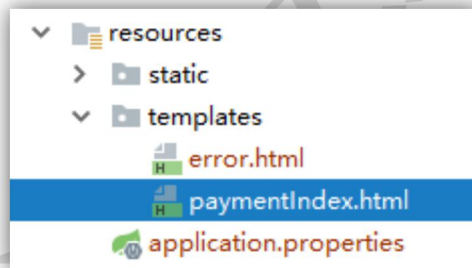
# gmall
192.168.67.163 payment.gmall.com cart.gmall.com pas.
```

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



订单提交成功，请尽快付款！订单号：12 应付金额14473.00元

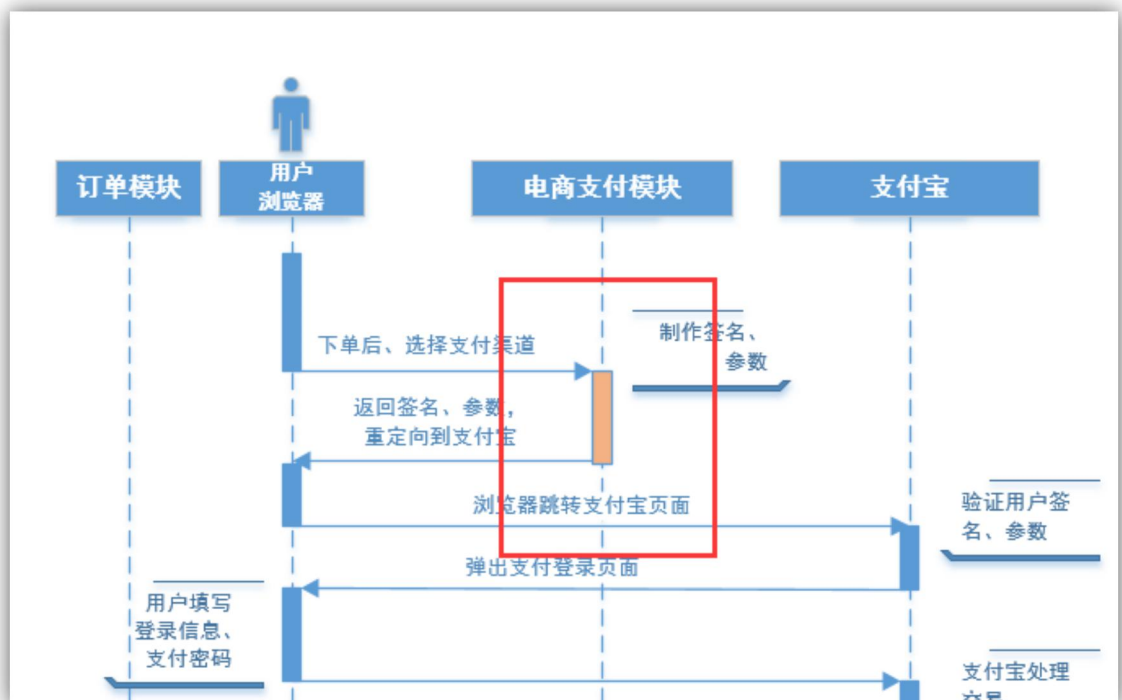
推荐使用 扫码支付请您在24小时内完成支付，否则订单会被自动取消(库存紧订单请参见详情页时限) [订单详情](#)

☒ 支付宝

☐ 微信支付

立即支付

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

关键参数说明:

配置参数	示例值解释	获取方式/示例值
URL	支付宝网关 (固定)	https://openapi.alipay.com/gateway.do
APPID	APPID 即创建应用后生成	获取见上面 创建应用
APP_PRIVATE_KEY	开发者私钥, 由开发者自己生成	获取详见上面 配置密钥
FORMAT	参数返回格式, 只支持json	json (固定)
CHARSET	编码集, 支持GBK/UTF-8	开发者根据实际工程编码配置
ALIPAY_PUBLIC_KEY	支付宝公钥, 由支付宝生成	获取详见上面 配置密钥
SIGN_TYPE	商户生成签名字符串所使用的签名算法类型, 目前支持RSA2和RSA, 推荐使用RSA2	RSA2

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

<p>电脑网站支付</p> <p>产品介绍</p> <p>快速接入</p> <p>支付结果异步通知</p> <p>SDK&Demo</p> <p>API列表</p> <p>联调问题排查</p>			<p>电脑网站支付API列表</p> <p>此列表包含该产品所涉及的所有接口, 点击“查看文档”可查看接口的公共请求参数, 业务请求参数, 返回参数, 其他语言请求示例以及错误码等。</p>		
接口英文名	接口中文名	API文档			
alipay.trade.page.pay	统一收单下单并支付页面接口	查看文档			
alipay.trade.refund	统一收单交易退款接口	查看文档			
alipay.trade.fastpay.refund.query	统一收单交易退款查询接口	查看文档			
alipay.trade.query	统一收单线下交易查询接口	查看文档			
alipay.trade.close	统一收单交易关闭接口	查看文档			
alipay.data.dataservice.bill.downloadurl.query	查询对账单下载地址	查看文档			

请求参数

参数	类型	是否必填	最大长度	描述	示例值
out_trade_no	String	是	64	商户订单号，64个字符以内、可包含字母、数字、下划线；需保证在商户端不重复	20150320010101001
product_code	String	是	64	销售产品码，与支付宝签约的产品码名称。 注： 目前仅支持FAST_INSTANT_TRADE_PAY	FAST_INSTANT_TRADE_PAY
total_amount	Price	是	11	订单总金额，单位为元，精确到小数点后两位，取值范围[0.01,1000000000]	88.88
subject	String	是	256	订单标题	Iphone6 16G

2 、 支付信息的保存

表结构 payment_info

payment_info(支付信息表)	
 id	编号
App out_trade_no	对外业务编号
App order_id	订单编号
App alipay_trade_no	支付宝交易编号
### total_amount	支付金额
App subject	交易内容
App payment_status	支付状态
 create_time	创建时间
 callback_time	回调时间
App 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
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 {  
  
    @Column  
    @Id  
    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 confirmTime;  
  
    @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 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
DefaultAlipayClient(alipay_url,app_id,app_private_key,format,charset,
alipay_public_key,sign_type );

    return alipayClient;
}
}
```

alipayClient=new

增加 alipay.properties

```
alipay_url=https://openapi.alipay.com/gateway.do

app_id=2018020102122556

app_private_key=MIIEvQI.....N1b/88D5yMuaZhZNFeUdWb+SmtP9DAzAW
YefJzetw/3cIimWu4NJzVQOaojaGA58oo2+fub43Xn25Jq4rvSVe3oLdb5xWkw5Q=

alipay_public_key=MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAhkZi6W0wn/prX
+NIIF9ATb5Z8ReKK4hFYtBrwe.....G+qbjTcZAzgNPfuiD0zXgt/YYjMQMzck75B0mwnYO
am2aj0DUSQn8Xybsa7wQIDAQAB

return_payment_url=http://payment.gmall.com/alipay/callback/return
```

```
notify_payment_url=http://60.205.215.91/alipay/callback/notify  
,  
return_order_url=http://order.gmall.com/list
```

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。

4 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();
}

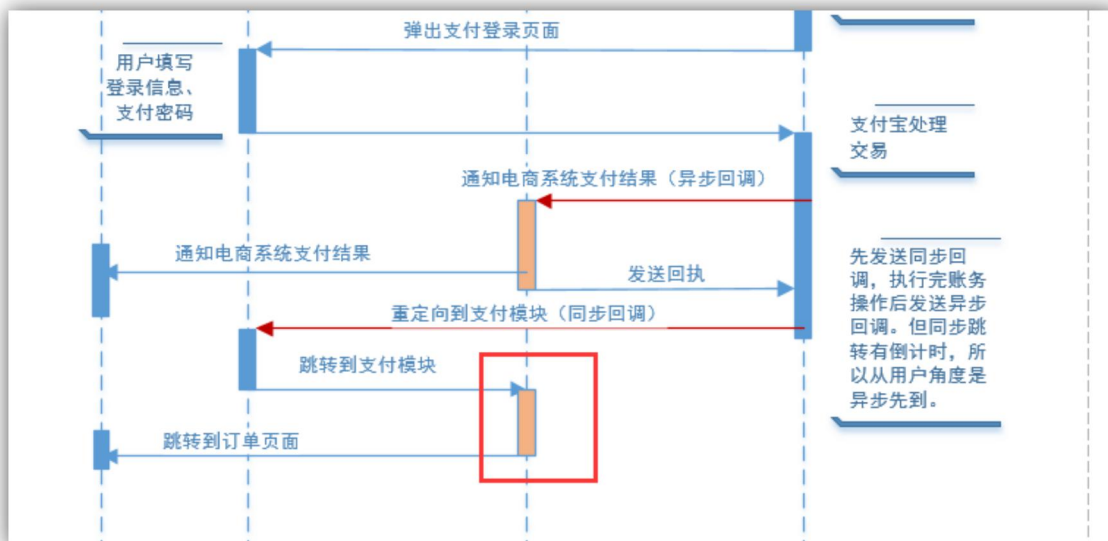
httpServletResponse.setContentType("text/html;charset=utf-8" );

//把表单 html 打印到客户端浏览器
return ResponseEntity.ok().body(form) ;
}
```

测试页面



五 支付后回调—同步回调



PaymentController

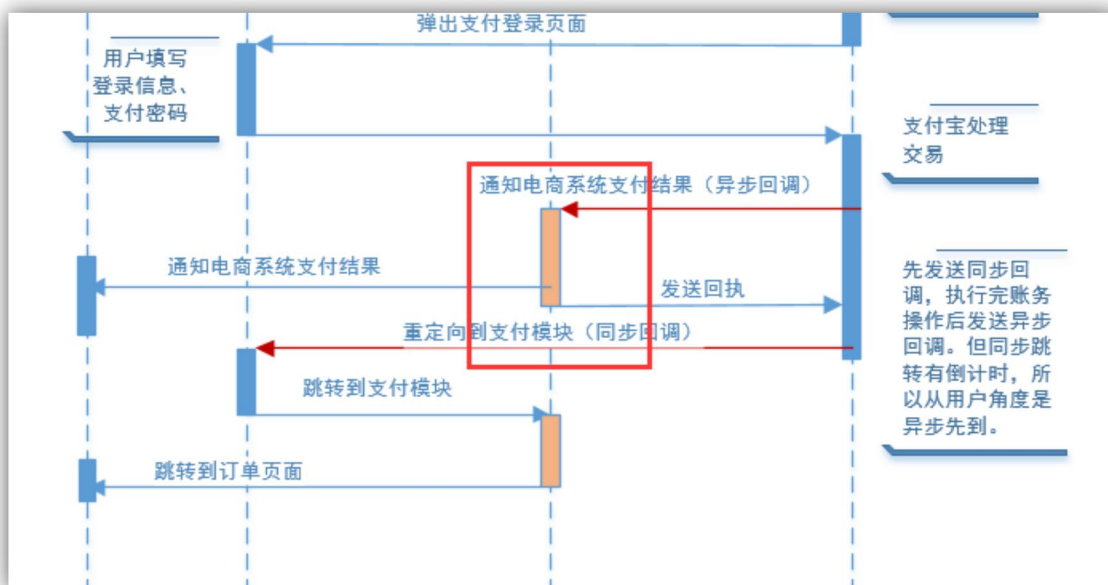
```

@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.getStatus()==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.getId());
        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。

```
-----callbackstart 支付宝开始回调[gmt_create=2018-04-01 17:57:22, charset=utf-8, gmt_payment=2018-04-01 17:57:
27, notify_time=2018-04-01 17:57:27, subject=小米6 全网通 6GB+128GB 亮蓝色 移动联通电信4G手机 双卡双待等6件商品, sig
n=dZr9EipGiroKUsMoYCd66yihWCIKnZzXEiQHhzn1JW/VjXYLAGZE4ds0Xu/99T5CrisJgtkTXqMhgP6MPk8Pv9cKbkC9mgR7Hp fhz9hyQtd/HSgRbU
njp19m7JehAwfNZtCi0eg76Qkm0S+8QNFeUp/PVKhr0zu0ZwyMQx4s0sx3AFzi6kI2be0UHEs2ad/c3HsT8GZU0K7MfPv5iMFgGmIovFJ+KMfkjrZ29N
jW5gEm8M/dsLFjMh4jiF3bfjE8w9PDopWsReN6TLgxfX0d8u0mZ2E3UXVCvSYdeKCicPGxMreKKeT7xAr7GLTvEo4YmvlICf1RrLmleMbDGlvnTTw==,
buyer_id=20880002039366201, invoice_amount=0.01, version=1.0, notify_id=e177aaaf16alcelcdd316690c6b02achjp, fund_bill
_list=[{"amount":"0.01","fundChannel":"ALIPAYACCOUNT"}], notify_type=trade status sync, out_trade_no=ATGUIGU15224685
15065000121, total_amount=0.01, trade_status=TRADE_SUCCESS, trade_no=2018040121001004200595857816, auth_app_id=20180
20102122556, receipt_amount=0.01, point_amount=0.00, app_id=2018020102122556, buyer_pay_amount=0.01, sign_type=RSA2,
seller_id=2088921750292524}
ATGUIGU1522468515065000121 -----验签通过! !
检查是否已处理= ATGUIGU1522468515065000121
未处理, 更新状态= ATGUIGU1522468515065000121
```