**支付宝开发**

# 第三方支付

## 在线支付

在线支付是指卖方与买方通过因特网上的电子商务网站进行交易时，银行为其提供网上资金结算服务的一种业务。它为企业和个人提供了一个安全、快捷、方便的电子商务应用环境和网上资金结算工具。在线支付不仅帮助企业实现了销售款项的快速归集，缩短收款周期，同时也为个人网上银行客户提供了网上消费支付结算方式，使客户真正做到足不出户，网上购物。

## 聚合支付

聚合支付：也称“融合支付”，是指只从事“支付、结算、清算”服务之外的“支付服务”，依托银行、非银机构或清算组织，借助银行、非银机构或清算组织的支付通道与清结算能力，利用自身的技术与服务集成能力，将一个以上的银行、非银机构或清算组织的支付服务，整合到一起，为商户提供包括但不限于“支付通道服务”、“集合对账服务”、“技术对接服务”、“差错处理服务”、“金融服务引导”、“会员账户服务”、“作业流程软件服务”、“运行维护服务”、“终端提供与维护”等服务内容，以此减少商户接入、维护支付结算服务时面临的成本支出，提高商户支付结算系统运行效率的，并收取增值收益的支付服务。

# 第三方支付流程

回调方式：同步回调、异步回调

回调场景: 告诉商户支付通知结果

同步回调: 整个支付流程完毕，使用同步方式将参数重定向给商户平台，一般场景用于展示结果。

异步回调: 第三方支付接口发一个后台通知给商户平台，一般场景用户修改订单信息。

# 数据安全加密

## 单向加密

单向加密算法主要用来验证数据传输的过程中，是否被篡改过。

MD5(Message Digest algorithm 5，信息摘要算法)

SHA(Secure Hash Algorithm，安全散列算法)

HMAC(Hash Message Authentication Code，散列消息鉴别码

## 对称加密

你可以这么理解，一方通过密钥将信息加密后，把密文传给另一方，另一方通过这个相同的密钥将密文解密，转换成可以理解的明文。他们之间的关系如下：

明文 <-> 密钥 <-> 密文

常用对称加密方案 DES、AES、Base64

## 非对称加密

在通信双方，如果使用非对称加密，一般遵从这样的原则：公钥加密，私钥解密。同时，一般一个密钥加密，另一个密钥就可以解密。

因为公钥是公开的，如果用来解密，那么就很容易被不必要的人解密消息。因此，**私钥也可以认为是个人身份的证明。**

如果通信双方需要互发消息，那么应该建立两套非对称加密的机制（即两对公私钥密钥对），发消息的一方使用对方的公钥进行加密，接收消息的一方使用自己的私钥解密。

就从上面提到的这个对称加密的缺点开始，怎么做到即时一个人的密钥被盗窃了，最起码保证你给其他人发送密文不被破解。于是，人们就想出了这么个办法，首先，我们停止分享共同的密钥，因为上面的 bug 就是来源于共享一个密钥，那么怎么办呢？每个人生成一个“私钥-公钥”对，这个私钥需要每个人自行进行保护！公钥可以随便分享，后面详细说，同时，生成的这个“私钥-公钥”对还有个强大的功能就是，使用私钥加密的信息，只能由该私钥对应的公钥才能解密，使用公钥加密的信息，只能由该公钥对应的私钥才能解密！

常用非对称加密 RSA 一般用于数据安全加密特别，支付领域。

## 参数验签

# 外网映射工具

## Ngrok使用

●windows用户:

1,下载windows版本的客户端，解压到你喜欢的目录  
2,在命令行下进入到path/to/windows\_386/下  
3,执行 ngrok -config=ngrok.cfg -subdomain xxx 80 //(xxx 是你自定义的域名前缀)

4,如果开启成功 你就可以使用 xxx.tunnel.qydev.com 来访问你本机的 127.0.0.1:80 的服务啦  
5,如果你自己有顶级域名，想通过自己的域名来访问本机的项目，那么先将自己的顶级域名解析到123.57.165.240(域名需要已备案哦)，然后执行./ngrok -config=ngrok.cfg -hostname xxx.xxx.xxx 80 //(xxx.xxx.xxx是你自定义的顶级域名)  
6，如果开启成功 你就可以使用你的顶级域名来访问你本机的 127.0.0.1:80 的服务啦  
7,如果失败 就加下交流群 反馈下问题 本屌会看看什么原因....吧

## Natapp使用

 windows ,点击开始->运行->命令行提示符 后进入 natapp.exe的目录  
    运行   natapp -authtoken= 175396706488ac93

# 支付宝开发环境

## 支付宝沙箱

蚂蚁沙箱环境(Beta)是协助开发者进行接口功能开发及主要功能联调的辅助环境。沙箱环境模拟了开放平台部分产品的主要功能和主要逻辑（当前沙箱支持产品请参考“沙箱支持产品列表”）。  
在开发者应用上线审核前，开发者可以根据自身需求，先在沙箱环境中了解、组合和调试各种开放接口，进行开发调通工作，从而帮助开发者在应用上线审核完成后，能更快速、更顺利的进行线上调试和验收工作。  
如何使用和配置沙箱环境请参考《沙箱环境使用说明》。

注意：

* 由于沙箱为模拟环境，在沙箱完成接口开发及主要功能调试后，请务必在蚂蚁正式环境进行完整的功能验收测试。所有返回码及业务逻辑以正式环境为准。
* 为保证沙箱稳定，沙箱环境测试数据会进行定期数据清理。Beta测试阶段每周日中午12点至每周一中午12点为维护时间。在此时间内沙箱环境部分功能可能会不可用，敬请谅解。
* 请勿在沙箱进行压力测试，以免触发相应的限流措施，导致无法正常使用沙箱环境。
* 沙箱支持的各个开放产品，沙箱使用的特别说明请参考各产品的快速接入文档或技术接入文档章节。

<https://openhome.alipay.com/platform/appDaily.htm>

## 环境准备

### 修改AlipayConfig信息

|  |
| --- |
| /\* \*  \*类名：AlipayConfig  \*功能：基础配置类  \*详细：设置帐户有关信息及返回路径  \*修改日期：2017-04-05  \*说明：  \*以下代码只是为了方便商户测试而提供的样例代码，商户可以根据自己网站的需要，按照技术文档编写,并非一定要使用该代码。  \*该代码仅供学习和研究支付宝接口使用，只是提供一个参考。  \*/  **public** **class** AlipayConfig {    //↓↓↓↓↓↓↓↓↓↓请在这里配置您的基本信息↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓  // 应用ID,您的APPID，收款账号既是您的APPID对应支付宝账号  **public** **static** String *app\_id* = "2016092100564758";    // 商户私钥，您的PKCS8格式RSA2私钥  **public** **static** String *merchant\_private\_key* = "";    // 支付宝公钥,查看地址：https://openhome.alipay.com/platform/keyManage.htm 对应APPID下的支付宝公钥。  **public** **static** String *alipay\_public\_key* = "MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDIgHnOn7LLILlKETd6BFRJ0GqgS2Y3mn1wMQmyh9zEyWlz5p1zrahRahbXAfCfSqshSNfqOmAQzSHRVjCqjsAw1jyqrXaPdKBmr90DIpIxmIyKXv4GGAkPyJ/6FTFY99uhpiq0qadD/uSzQsefWo0aTvP/65zi3eof7TcZ32oWpwIDAQAB";  // 服务器异步通知页面路径 需http://格式的完整路径，不能加?id=123这类自定义参数，必须外网可以正常访问  **public** **static** String *notify\_url* = "http://itmayiedu.tunnel.qydev.com/notify\_url.jsp";  // 页面跳转同步通知页面路径 需http://格式的完整路径，不能加?id=123这类自定义参数，必须外网可以正常访问  **public** **static** String *return\_url* = "http://itmayiedu.tunnel.qydev.com/return\_url.jsp";  // 签名方式  **public** **static** String *sign\_type* = "RSA";    // 字符编码格式  **public** **static** String *charset* = "utf-8";    // 支付宝网关  **public** **static** String *gatewayUrl* = "https://openapi.alipaydev.com/gateway.do";    // 支付宝网关  **public** **static** String *log\_path* = "C:\\";  //↑↑↑↑↑↑↑↑↑↑请在这里配置您的基本信息↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑  /\*\*  \* 写日志，方便测试（看网站需求，也可以改成把记录存入数据库）  \* **@param** sWord 要写入日志里的文本内容  \*/  **public** **static** **void** logResult(String sWord) {  FileWriter writer = **null**;  **try** {  writer = **new** FileWriter(*log\_path* + "alipay\_log\_" + System.*currentTimeMillis*()+".txt");  writer.write(sWord);  } **catch** (Exception e) {  e.printStackTrace();  } **finally** {  **if** (writer != **null**) {  **try** {  writer.close();  } **catch** (IOException e) {  e.printStackTrace();  }  }  }  }  } |

### 运行Demo项目

# 项目使用集成支付宝

### 支付数据库表结构

|  |
| --- |
| CREATE TABLE `payment\_info` (  `id` int(11) NOT NULL AUTO\_INCREMENT,  `userid` int(11) DEFAULT NULL,  `typeid` int(2) DEFAULT NULL,  `orderid` varchar(50) DEFAULT NULL,  `price` decimal(10,0) DEFAULT NULL,  `source` varchar(10) DEFAULT NULL,  `state` int(2) DEFAULT NULL,  `created` datetime DEFAULT NULL,  `updated` datetime DEFAULT NULL,  `platformorderid` varchar(100) DEFAULT NULL,  PRIMARY KEY (`id`)  ) ENGINE=InnoDB AUTO\_INCREMENT=7 DEFAULT CHARSET=utf8; |

## 创建itmayiedu-shopp-pay服务工程

### application配置文件

|  |
| --- |
| **server:**  **port: 8768**  **# context-path: /member**  **eureka:**  **client:**  **serviceUrl:**  **defaultZone: http://localhost:8761/eureka/**  **spring:**  **application:**  **name: pay**  **redis:**  **host: 47.100.50.60**  **password: 123**  **port: 6379**  **pool:**  **max-idle: 100**  **min-idle: 1**  **max-active: 1000**  **max-wait: -1**  **#数据库连接信息**  **datasource:**  **name: test**  **url: jdbc:mysql://127.0.0.1:3306/itmayiedu-pay**  **username: root**  **password: root**  **# 使用druid数据源**  **type: com.alibaba.druid.pool.DruidDataSource**  **driver-class-name: com.mysql.jdbc.Driver**  **filters: stat**  **maxActive: 20**  **initialSize: 1**  **maxWait: 60000**  **minIdle: 1**  **timeBetweenEvictionRunsMillis: 60000**  **minEvictableIdleTimeMillis: 300000**  **validationQuery: select 'x'**  **testWhileIdle: true**  **testOnBorrow: false**  **testOnReturn: false**  **poolPreparedStatements: true**  **maxOpenPreparedStatements: 20** |

### Maven依赖

|  |
| --- |
| <dependencies>  <dependency>  <groupId>com.itmayiedu</groupId>  <artifactId>itmayiedu-shopp-api-pay</artifactId>  <version>0.0.1-SNAPSHOT</version>  </dependency>  <!-- springboot整合mybatis -->  <dependency>  <groupId>org.mybatis.spring.boot</groupId>  <artifactId>mybatis-spring-boot-starter</artifactId>  <version>1.3.0</version>  </dependency>  <!-- https://mvnrepository.com/artifact/com.github.1991wangliang/alipay-sdk -->  <dependency>  <groupId>com.github.1991wangliang</groupId>  <artifactId>alipay-sdk</artifactId>  <version>1.0.0</version>  </dependency>  </dependencies> |

### 数据库访问层

|  |
| --- |
| @Mapper  **public** **interface** PaymentInfoDao {  @Select("select \* from payment\_info where id=#{id}")  **public** PaymentInfo getPaymentInfo(@Param("id") Long id);  @Insert("insert into payment\_info ( id,userid,typeid,orderid,platformorderid,price,source,state,created,updated) value(null,#{userId},#{typeId},#{orderId},#{platformorderId},#{price},#{source},#{state},#{created},#{updated})")  @Options(useGeneratedKeys = **true**, keyProperty = "id") // 添加该行，product中的id将被自动添加  **public** Integer savePaymentType(PaymentInfo paymentInfo);  @Select("select \* from payment\_info where orderId=#{orderId}")  **public** PaymentInfo getByOrderIdPayInfo(@Param("orderId") String orderId);  @Update("update payment\_info set state =#{state},payMessage=#{payMessage},platformorderId=#{platformorderId},updated=#{updated} where orderId=#{orderId} ")  **public** **void** updatePayInfo(PaymentInfo paymentInfo);  } |

### 业务逻辑层

|  |
| --- |
| @Service  **public** **class** AliBaBaManagerImpl **implements** PayManager {  @Override  **public** String payInfo(PaymentInfo paymentInfo) **throws** AlipayApiException {  //获得初始化的AlipayClient  AlipayClient alipayClient = **new** DefaultAlipayClient(AlipayConfig.*gatewayUrl*, AlipayConfig.*app\_id*, AlipayConfig.*merchant\_private\_key*, "json", AlipayConfig.*charset*, AlipayConfig.*alipay\_public\_key*, AlipayConfig.*sign\_type*);    //设置请求参数  AlipayTradePagePayRequest alipayRequest = **new** AlipayTradePagePayRequest();  alipayRequest.setReturnUrl(AlipayConfig.*return\_url*);  alipayRequest.setNotifyUrl(AlipayConfig.*notify\_url*);    //商户订单号，商户网站订单系统中唯一订单号，必填  String out\_trade\_no =paymentInfo.getOrderId();  //付款金额，必填  String total\_amount = paymentInfo.getPrice()+"";  //订单名称，必填  String subject ="蚂蚁课堂会员费用";  // //商品描述，可空  // String body = new String(request.getParameter("WIDbody").getBytes("ISO-8859-1"),"UTF-8");    alipayRequest.setBizContent("{\"out\_trade\_no\":\""+ out\_trade\_no +"\","  + "\"total\_amount\":\""+ total\_amount +"\","  + "\"subject\":\""+ subject +"\","  + "\"product\_code\":\"FAST\_INSTANT\_TRADE\_PAY\"}");    //若想给BizContent增加其他可选请求参数，以增加自定义超时时间参数timeout\_express来举例说明  //alipayRequest.setBizContent("{\"out\_trade\_no\":\""+ out\_trade\_no +"\","  // + "\"total\_amount\":\""+ total\_amount +"\","  // + "\"subject\":\""+ subject +"\","  // + "\"body\":\""+ body +"\","  // + "\"timeout\_express\":\"10m\","  // + "\"product\_code\":\"FAST\_INSTANT\_TRADE\_PAY\"}");  //请求参数可查阅【电脑网站支付的API文档-alipay.trade.page.pay-请求参数】章节    //请求  String result = alipayClient.pageExecute(alipayRequest).getBody();  **return** result;  }  } |

### 服务实现层

|  |
| --- |
| @RestController  **public** **class** PayServiceImpl **extends** BaseApiService **implements** PayService {  @Autowired  **private** PaymentInfoDao paymentInfoDao;  @Autowired  **private** AliBaBaManagerImpl aliBaBaManagerImpl;  @Override  **public** ResponseBase getPayToken(@RequestBody PaymentInfo paymentInfo) {  // 1.插入参数提交信息  Integer savePaymentType = paymentInfoDao.savePaymentType(paymentInfo);  **if** (savePaymentType <= 0) {  **return** setResultError("系统错误!");  }  // 2.生成对应token  String payToken = TokenUtils.*getPayToken*();  // 3.存放在redis中  Integer payId = paymentInfo.getId();  baseRedisService.setString(payToken, payId + "", Constants.***PAY\_TOKEN\_MEMBER\_TIME***);  // 4.返回token給消費者  JSONObject result = **new** JSONObject();  result.put("payToken", payToken);  **return** setResultSuccess(result);  }  @Override  **public** ResponseBase payInfo(String payToken) {  **if** (StringUtils.*isEmpty*(payToken)) {  **return** setResultError("token不能为空!");  }  String payId = (String) baseRedisService.getString(payToken);  **if** (StringUtils.*isEmpty*(payId)) {  **return** setResultError("支付已经超时!");  }  PaymentInfo paymentInfo = paymentInfoDao.getPaymentInfo(Long.*parseLong*(payId));  **if** (paymentInfo == **null**) {  **return** setResultError("未找到交易类型.");  }  // 判断类型 调用 具体业务接口  Long typeId = paymentInfo.getTypeId();  PayManager payManager = **null**;  // 调用支付接口  **if** (typeId == 1) {  payManager = aliBaBaManagerImpl;  }  **try** {  String payInfo = payManager.payInfo(paymentInfo);  JSONObject payInfoJSON = **new** JSONObject();  payInfoJSON.put("payInfo", payInfo);  **return** setResultSuccess(payInfoJSON);  } **catch** (AlipayApiException e) {  **return** setResultError("支付错误!");  }  }  } |

### 项目启动

|  |
| --- |
| @SpringBootApplication  @EnableEurekaClient  **public** **class** PayApp {  **public** **static** **void** main(String[] args) {  SpringApplication.*run*(PayApp.**class**, args);  }    } |

## 创建itmayiedu-shopp-pc-web

|  |
| --- |
| @Controller  **public** **class** PayController {  @Autowired  **private** PayServiceFegin payServiceFegin;  @RequestMapping("/pay")  **public** **void** pay(String payToken, HttpServletResponse resp) **throws** IOException {  resp.setContentType("text/html;charset=utf-8");  ResponseBase payInfo = payServiceFegin.payInfo(payToken);  PrintWriter out = resp.getWriter();  **if** (!payInfo.getRtnCode().equals(Constants.***HTTP\_RES\_CODE\_200***)) {  out.println("ERROR");  **return**;  }  LinkedHashMap mapPayInfo = (LinkedHashMap) payInfo.getData();  String payInfoHtml = (String) mapPayInfo.get("payInfoHtml");  out.println(payInfoHtml);  out.close();  }  } |

# Web常用工具网站

Html在线格式地址