```
  private String appPrivateKey;
    /**
     * 支付宝公钥，由支付宝生成。
     */
    private String alipayPublicKey;
    /**
     * 用户确认支付后，支付宝调用的页面返回路径
     */
    private String returnUrl;
    /**
     * 支付宝服务器主动通知商户服务器里的异步通知回调（需要公网能访问）
     */
    private String notifyUrl;
    /**
     * 参数返回格式，只支持JSON
     */
    private String format = "JSON";
    /**
     * 请求使用的编码格式
     */
    private String charset = "UTF-8";
    /**
     * 生成签名字符串所使用的签名算法类型
     */
    private String signType = "RSA2";
}
```

4）新建支付宝请求客户端配置类`AlipayClientConfig`，用于向支付宝API发送请求

```
@Configuration
public class AlipayClientConfig {

    @Bean
    public AlipayClient alipayClient(AlipayConfig config){
        return new
DefaultAlipayClient(config.getGatewayUrl(),config.getAppId(),config.getA
ppPrivateKey(),
config.getFormat(),config.getCharset(),config.getAlipayPublicKey(),config
.getSignType());
```

```
    }
}
```

订单管理接口实现

1）首先创建`alipay_order`表，用于存储订单信息

```
CREATE TABLE `alipay_order` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT,
  `order_id` varchar(64) NOT NULL COMMENT '订单ID',
  `subject` varchar(255) DEFAULT NULL COMMENT '订单标题/商品标题/交易标题',
  `total_amount` decimal(10,2) DEFAULT NULL COMMENT '订单总金额',
  `trade_status` varchar(255) DEFAULT NULL COMMENT '交易状态',
  `trade_no` varchar(255) DEFAULT NULL COMMENT '支付宝交易号',
  `buyer_id` varchar(255) DEFAULT NULL COMMENT '买家支付宝账号',
  `gmt_payment` datetime DEFAULT NULL COMMENT '交易付款时间',
  `buyer_pay_amount` decimal(10,2) DEFAULT NULL COMMENT '用户在交易中支付的金额',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=9 DEFAULT CHARSET=utf8mb4 COMMENT='支付宝支付订单表';
```

2）新建一个`AlipayOrder`实体类用于映射`alipay_order`表中的数据行

```
package com.macro.mall.model;

import com.fasterxml.jackson.annotation.JsonFormat;
import lombok.Getter;
import lombok.Setter;

import java.io.Serializable;
import java.math.BigDecimal;
import java.util.Date;

@Getter
```

```
@Setter
public class AlipayOrder implements Serializable {

    private Long id;
    /**
     * 订单ID
     */
    private String orderId;
    /**
     * 订单标题/商品标题/交易标题
     */
    private String subject;

    /**
     * 订单总金额
     */
    private BigDecimal totalAmount;

    /**
     * 交易状态
     */
    private String tradeStatus;

    /**
     * 支付宝交易号
     */
    private String tradeNo;
    /**
     * 买家支付宝账号
     */
    private String buyerId;
    /**
     * 交易付款时间，固定日期格式，使用东八区时间
     */
    @JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss", timezone =
"GMT+8")
    private Date gmtPayment;

    /**
     * 用户在交易中支付的金额
     */
    private BigDecimal buyerPayAmount;

}
```

3）创建支付宝订单管理控制器`AlipayOrderController`类

```
package com.macro.mall.portal.controller;


import com.macro.mall.common.api.CommonResult;
import com.macro.mall.model.AlipayOrder;
import com.macro.mall.portal.service.AlipayOrderService;
import io.swagger.annotations.Api;
import io.swagger.annotations.ApiOperation;
import io.swagger.v3.oas.annotations.tags.Tag;
import org.springframework.web.bind.annotation.*;

import javax.annotation.Resource;

@RestController
@RequestMapping("/alipayOrder")
@Api(tags = "AlipayOrderController")
@Tag(name = "AlipayOrderController", description = "支付宝订单管理")
public class AlipayOrderController {

    @Resource
    private  AlipayOrderService alipayOrderService;

    @ApiOperation("创建订单")
    @PostMapping("/createOrder")
    public CommonResult<AlipayOrder> createOrder(){
        AlipayOrder order = alipayOrderService.createOrder();
        return CommonResult.success(order);
    }

    @ApiOperation("查询订单")
    @GetMapping("/order")
    public CommonResult<AlipayOrder>
getOrderInfo(@RequestParam("orderId") String orderId){
        AlipayOrder order = alipayOrderService.getOrderInfo(orderId);
        return CommonResult.success(order);
    }
}
```

4）创建支付宝订单管理`AlipayOrderService`接口及实现类。

```
package com.macro.mall.portal.service.impl;

import cn.hutool.core.util.RandomUtil;
import com.macro.mall.mapper.AlipayOrderMapper;
import com.macro.mall.model.AlipayOrder;
import com.macro.mall.portal.service.AlipayOrderService;
import lombok.extern.slf4j.Slf4j;
import org.springframework.stereotype.Service;

import javax.annotation.Resource;
import java.math.BigDecimal;
import java.text.SimpleDateFormat;
import java.util.Date;

@Slf4j
@Service
public class AlipayOrderServiceImpl implements AlipayOrderService {
    @Autowired
    private AlipayOrderMapper alipayOrderMapper;

    @Override
    public AlipayOrder createOrder() {
        String orderId = new SimpleDateFormat("yyyyMMdd").format(new
Date()) + System.currentTimeMillis();
        AlipayOrder alipayOrder = new AlipayOrder();
        alipayOrder.setOrderId(orderId);
        //模拟数量
        int quantity = RandomUtil.randomInt(1, 10);
        BigDecimal price = new BigDecimal(20);
        alipayOrder.setSubject("测试商品"+quantity+"个");
        BigDecimal totalAmount = price.multiply(new BigDecimal(quantity));
        alipayOrder.setTotalAmount(totalAmount);
        alipayOrder.setGmtPayment(new Date());
        alipayOrder.setBuyerPayAmount(totalAmount);
        alipayOrderMapper.addOrder(alipayOrder);
        return alipayOrder;
    }

     @Override
    public AlipayOrder getOrderInfo(String orderId) {
        log.info("orderId={}", orderId);
      FINISHED（交易结束，不可退款）
            if(response.isSuccess()){
                return response.getTradeStatus();
            } else {
                return  response.getSubCode();
            }
```

```
        } else {
            log.error("查询支付宝账单失败！");
            return "TRADE_ERROR";
        }

    }
}
```

改mall-gateway 微服务项目

因为配置的内网穿透映射的内网服务端口为80，所以需要将`mall-gateway`项目中`application.yml`文件中的`server.port`值改为80，同时通过防火墙开放80端口服务

```
server:
  port: 80
```

`mall-gateway`网关项目用了`Spring-Security`来做安全认证和鉴权，由于暂时没有写前端页面，为了方便我们直接在浏览器中通过Get请求提交支付成功后页面能弹出支付宝的支付界面，我们需要对这次在`mall-portal`项目中添加的与创建订单和支付相关的接口全部先免鉴权。

```
secure:
  ignore:
    urls:
      - "/mall-portal/alipay/**"
      - "/mall-portal/alipayOrder/**"
```

功能测试
====

启动服务并诊断内网穿透客户端

启动`mall-swarm`项目的两个微服务`mall-gateway`、`和`mall-portal`和服务前需要子在本地先启动`Mysql`、`redis`、`rabbitmq`、`naocs`和`Mongodb`等服务，然后再依次启动`mall-gateway`` 和`mall-portal `两个微服务。

启动微服务成功后在自己的电脑桌面登录**花生壳客户端**，开启自定义诊断，出现如下诊断信息显示为连接成功的情况表示内网穿透服务开启成功。

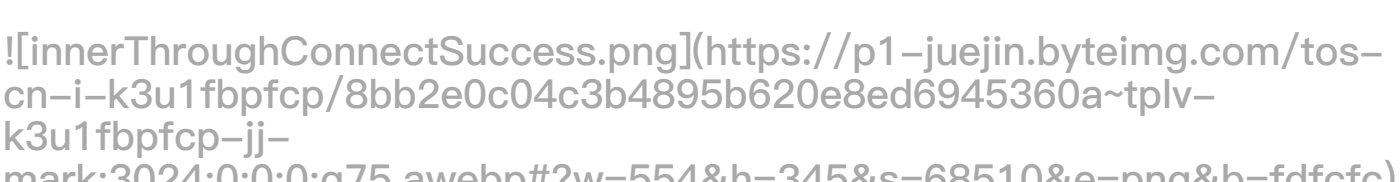

图 8 花生壳客户端应用内网穿透连接测试

支付效果测试

1）在ApiPost中调用创建订单接口，获取订单号和 支付金额等数据

```
POST https://ss8971sw6958.vicp.fun/mall-portal/alipayOrder/createOrder
// 接口返回数据
{
”code”: 200,
”message”: ”操作成功”,
”data”: {
”orderId”: ”202404091712673177214”,
”subject”: ”测试商品2个”,
”totalAmount”: 40,
”gmtPayment”: ”2024-04-09 22:32:57”,
”buyerPayAmount”: 40
}
}
```

未提交支付前根据order_id=’202404091712673177214’ 从

`alipay_order`表中查询出来的订单数据订单状态order_status字段为空

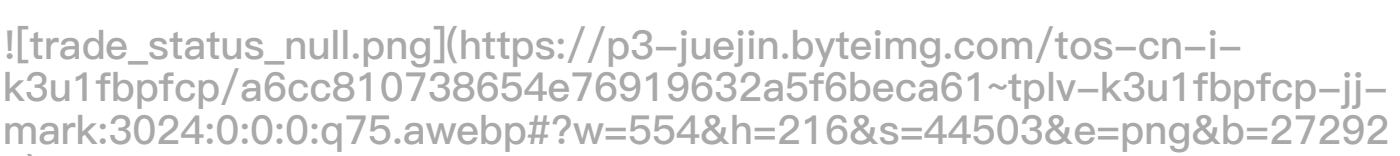

图 8 未提交支付前本地数据库订单表订单数据

2）将创建订单接口的返回参数作为支付接口的入参在谷歌浏览器中调用支付接口

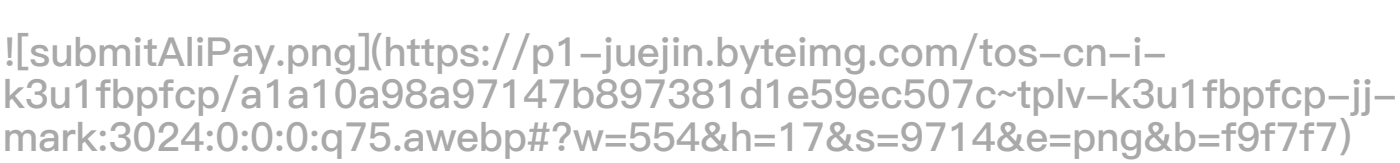

图 9 浏览器中输入提交订单支付链接

回车后返回支付宝支付界面

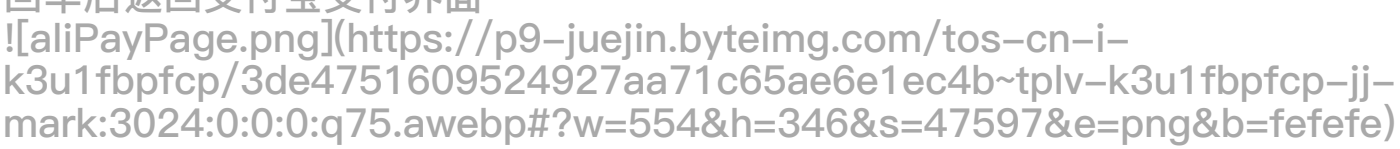

图 10 支付宝弹出的订单支付信息界面

3）然后进入沙箱支付测试账号买家信息页面查看完整的账户名和支付密码后输入后点击【下一步】进入支付确认页面。

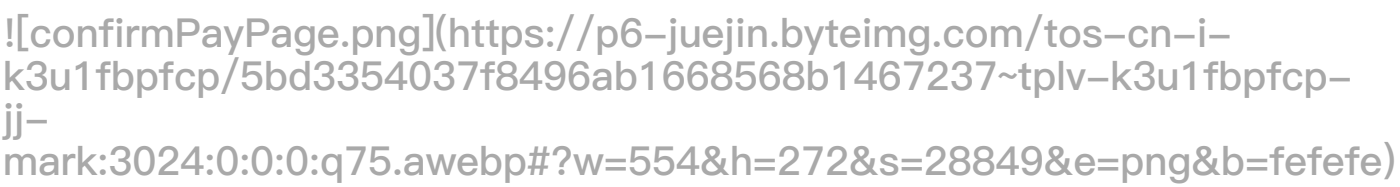

图 11 支付宝订单支付确认界面

4）点击确认付款后页面弹出付款成功信息

![pay_success.png](https://p3-juejin.byteimg.com/tos-cn-i-k3u1fbpfcp/bb6126c0937e4c719f5a461e6dd59a6c~tplv-k3u1fbpfcp-jj-mark:3024:0:0:0:q75.awebp#?w=554&h=166&s=31011&e=png&b=fefadd)

图 12 支付宝付款成功消息提示界面

5）支付成功后我们可以在mall-portal服务的控制台中根据关键字 `notify_params` 查到对应的支付宝对mall-portal服务的支付结果回调参数

```
2024-04-09 22:43:25 [http-nio-8085-exec-6] INFO c.m.m.p.s.impl.AlipayServiceImpl - notify_params={"gmt_create":"2024-04-09 22:42:44","charset":"UTF-8","gmt_payment":"2024-04-09 22:43:22","notify_time":"2024-04-09 22:43:24","subject":"测试商品2个","sign":"PSVSnQL6O8veDDBBlBctgAiBUWY8pa3l9RDcWRRCeer6Nkvos/4F2hiIm49X2s5H3RIbhl+lWquG7CmYurCKfBKvVgaIlWakJGBooY52sYDYWE95lBkFo6x59aZ+erF3lpX8EsFXKFiBuuJJytT91Wy2lAlAcTRaYnldMaOTd6AxqJ1Pabc0MSSzdUeSRFlXnd3dbLOX8lZyANLZOjE6sqiLYS9Xp2Co6UvDJNJltxhakc8fEQE3VRi8CTrzce99qxM8kLb0iyyMgcd4EykDwxR65Yn5K7mmNticbh9D0FqhWPCdH8Dvt2yAwY3GjQlGvsz/EJ3JcjFnQWd2f3mwow==","buyer_id":"2088722032870579","invoice_amount":"40.00","version":"1.0","notify_id":"2024040901222224323070570502795855","fund_bill_list":"[{\"amount\":\"40.00\",\"fundChannel\":\"ALIPAYACCOUNT\"}]","notify_type":"trade_status_sync","out_trade_no":"2024040917126731777214","total_amount":"40.00","trade_status":"TRADE_SUCCESS","trade_no":"2024040922001470570502624165","auth_app_id":"9021000135678698","receipt_amount":"40.00","point_amount":"0.00","buyer_pay_amount":"40.00","app_id":"9021000135678698","sign_type":"RSA2","seller_id":"2088721032870561"}
```

6）再次根据`orderId`的值执行查询支付订单的sql脚本可以看到 trade_status、trade_no 和 buyer_id 等三个之前为null 的字段都已更新

![update_trade_status.png](https://p9-juejin.byteimg.com/tos-cn-i-k3u1fbpfcp/d74b26a4e4fb4dc0bcc750fa45d5b1a7~tplv-k3u1fbpfcp-jj-mark:3024:0:0:0:q75.awebp#?w=554&h=165&s=38349&e=png&b=27292c)

图 13 提单提交支付成功后本地数据库订单表数据变化

trade_status字段值为 TRADE_SUCCESS 表示交易成功。

7）查询支付结果接口

如果没有开通内网穿透服务，那么我们就只能通过调用查询支付结果查询订单的交易状态

在ApiPost中调用以下GET接口

```
GET https://ss8971sw6958.vicp.fun/mall-portal/alipay/queryOrder?outTradeNo=202404091712673177214
// 返回响应信息
{
”code”: 200,
”message”: ”操作成功”,
”data”: ”TRADE_SUCCESS”
}
```

响应数据中的data字段值就是订单支付后的交易状态值。

总结
==

本文通过集成支付宝支付，在mall微服务项目中实现了一套支付流程，使用内网穿透接收到订单支付后来自阿里支付结果的回调数据。这里也仅仅只是实现了在线支付，支付里面的场景还是比较多的，包括扫码支付、当面付和申请退款等场景，只不过只是接口和参数不同而已。

其实支付功能并不难实现，支付宝沙箱环境让我们无需复杂的商业流程，就可以在沙箱环境快速实现支付功能。但是打通线上正式支付的场景时还是需要我们在支付宝开发平台创建对应的网站应用或小程序并开通产品，然后提交营业执照等材料申请支付宝签约。而如果没有营业执照的前提下仍然想要开通个人应用的支付功能，可以[YunGouOS平台](http://cxyroad.com/

”https://www.yungouos.com/#/alipay/apply”)开通服务商模式签约实现应用的支付功能。

本文首发个人公众号【阿福谈Web编程】，有不懂的地方可以通过我的公众号，然后点击【联系作者】菜单可以获得笔者的个人联系方式，大家一起交流进步。

原文链接: https://juejin.cn/post/7363549357410353191