

[🏠](#) / [开发指南](#) / [支付手册](#)[芋道源码](#) [📅 2023-07-09](#)

支付宝支付接入

0. 概述

在 `yudao-module-pay-biz` 模块的 `demo` 模块，我们提供了一个 **支付** 接入的示例。

它支持如下支付渠道：

- 支付宝 [电脑网站](#) 支付
- 支付宝 [手机网站](#) 支付
- 支付宝 [当面付](#) (条码支付)
- 支付宝 [扫码](#) 支付
- 微信 [付款码](#) 支付
- 微信 [Native](#) 支付

疑问：为什么不支持微信小程序、公众号支付？

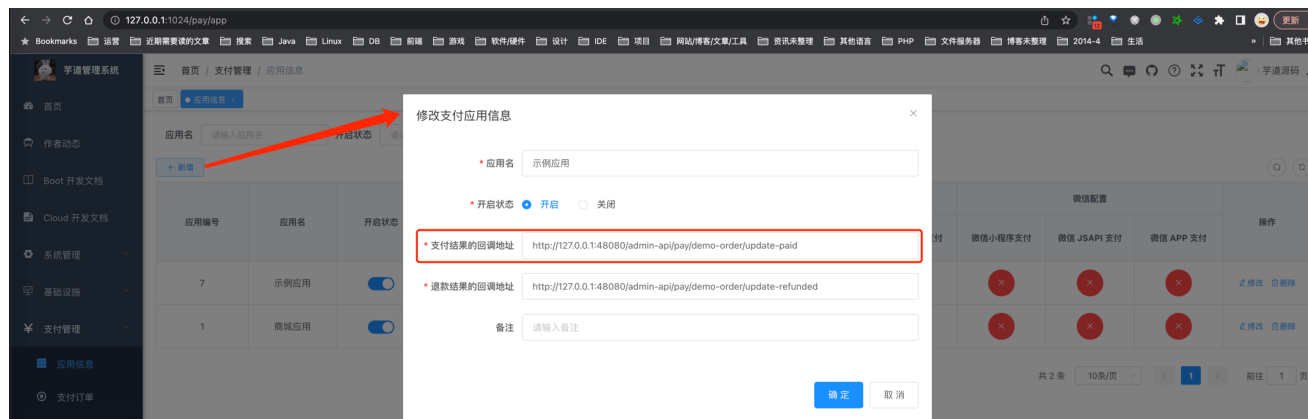
因为这 2 种微信支付方式，只能在微信环境中进行，而我们的接入示例使用 PC 浏览器，所以无法进行。

你可以阅读 [微信小程序支付接入](#) 和 [微信公众号支付接入](#) 文档，进行相关的支付接入。

下面，我们以 `demo` 模块为例子，讲解如何接入支付宝的 [电脑网站](#) 支付。

1. 第一步，配置支付渠道

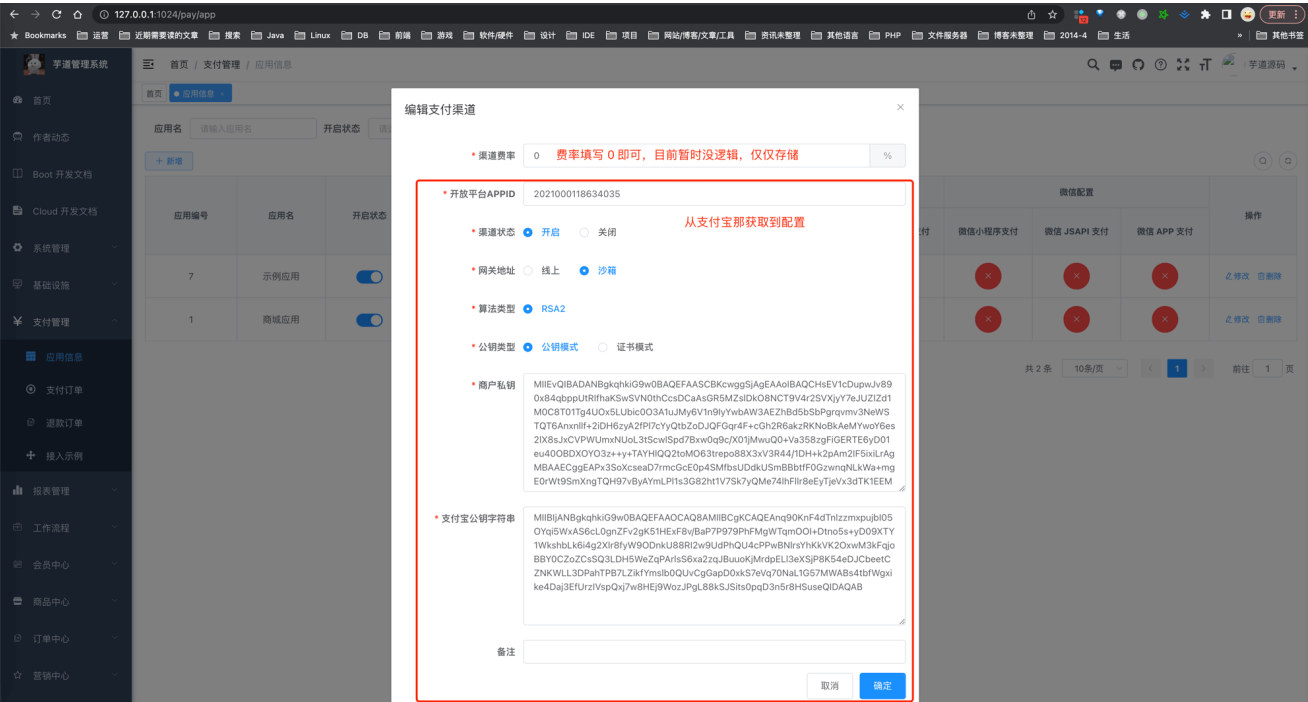
① 在 [支付管理 -> 应用信息] 菜单，新建 `demo` 模块使用的支付应用。如下图所示：



- 支付结果的回调地址：对应「2.4 第四步，实现回调接口」实现的接口。这里因为我们在本地演示，所以使用了 `127.0.0.1` IP 地址，如果你部署到服务器上，要设置服务器可调用的 IP 地址。

因为 demo 模块的支付应用默认已经创建，所以你可以直接使用。注意，这里的应用编号“7”稍后会使用到。

② 点击“示例用户”对应的【支付宝 PC 网站支付】，进入支付渠道的配置。如下图所示：



如果你还没有支付宝开放平台的账号，可以先参考《沙箱环境》文档，申请一个测试账号，我目前就是这么测试的。只需要阅读该文档的如下小节即可：



2. 第二步，实现支付调用【重要】

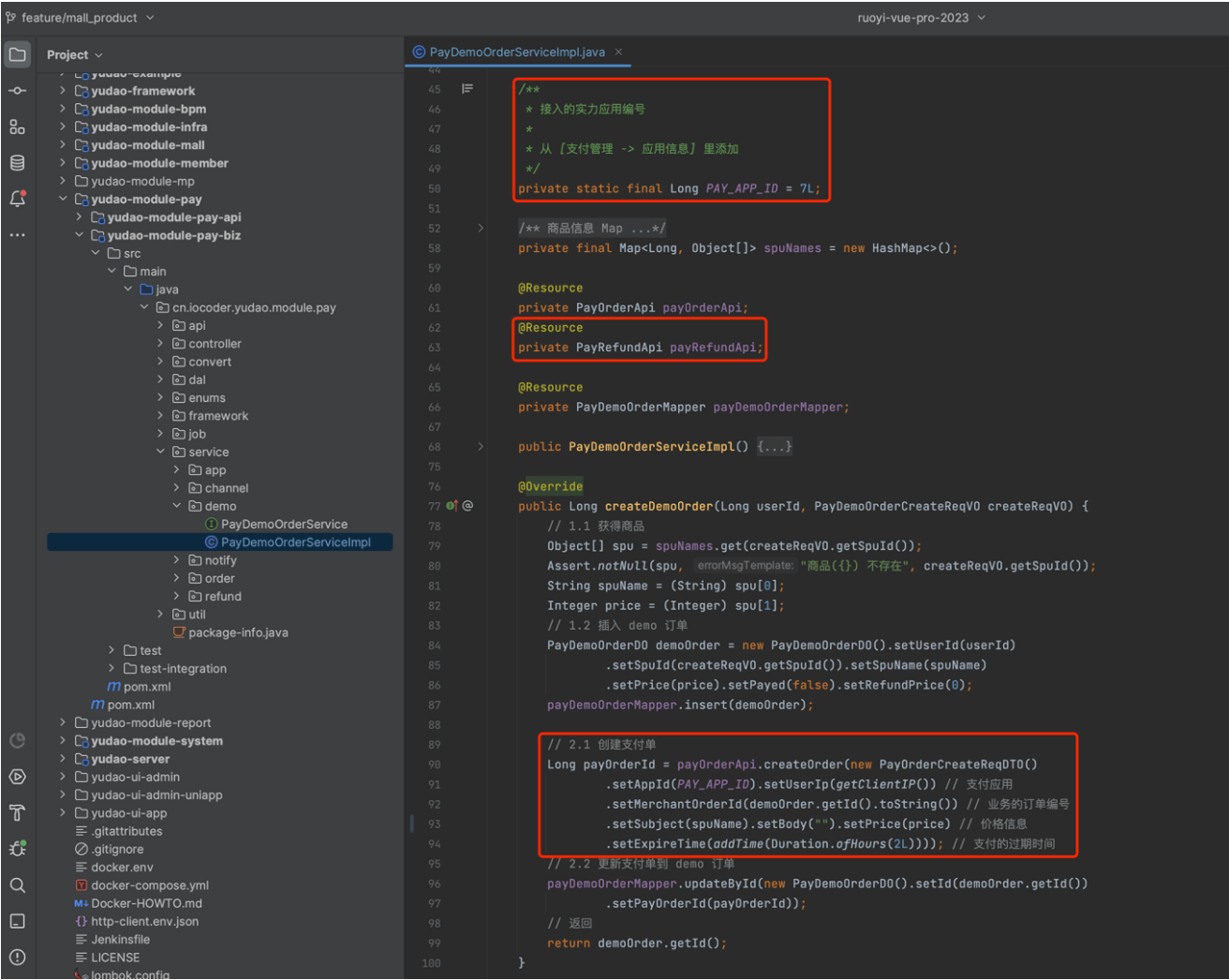
友情提示：由于 demo 模块的支付接入已经实现，这里你只要看懂什么意思即可，不用操作。

① 【后端】在 demo 模块所在的 yudao-module-xx-biz 模块的 pom.xml 文件，引入 yudao-module-pay-api 依赖，这样才可以调用到 PayOrderApi 接口。代码如下：

```
<dependency>
<groupId>cn.iocoder.boot</groupId>
```

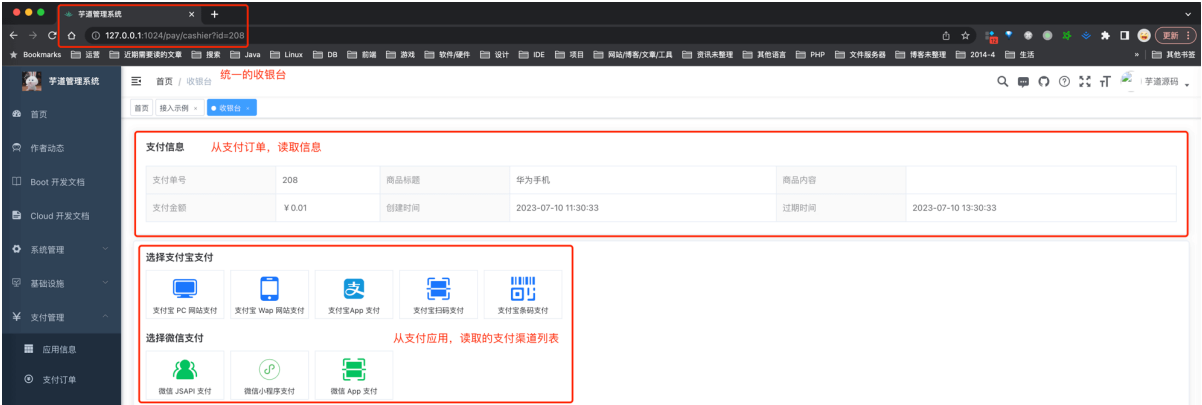
```
<artifactId>yudao-module-pay-api</artifactId>
<version>${revision}</version>
</dependency>
```

② 【后端】在 demo 模块的下单逻辑中，需要调用 PayOrderApi 的 #createOrder(...) 方法，创建支付单。如下图所示：

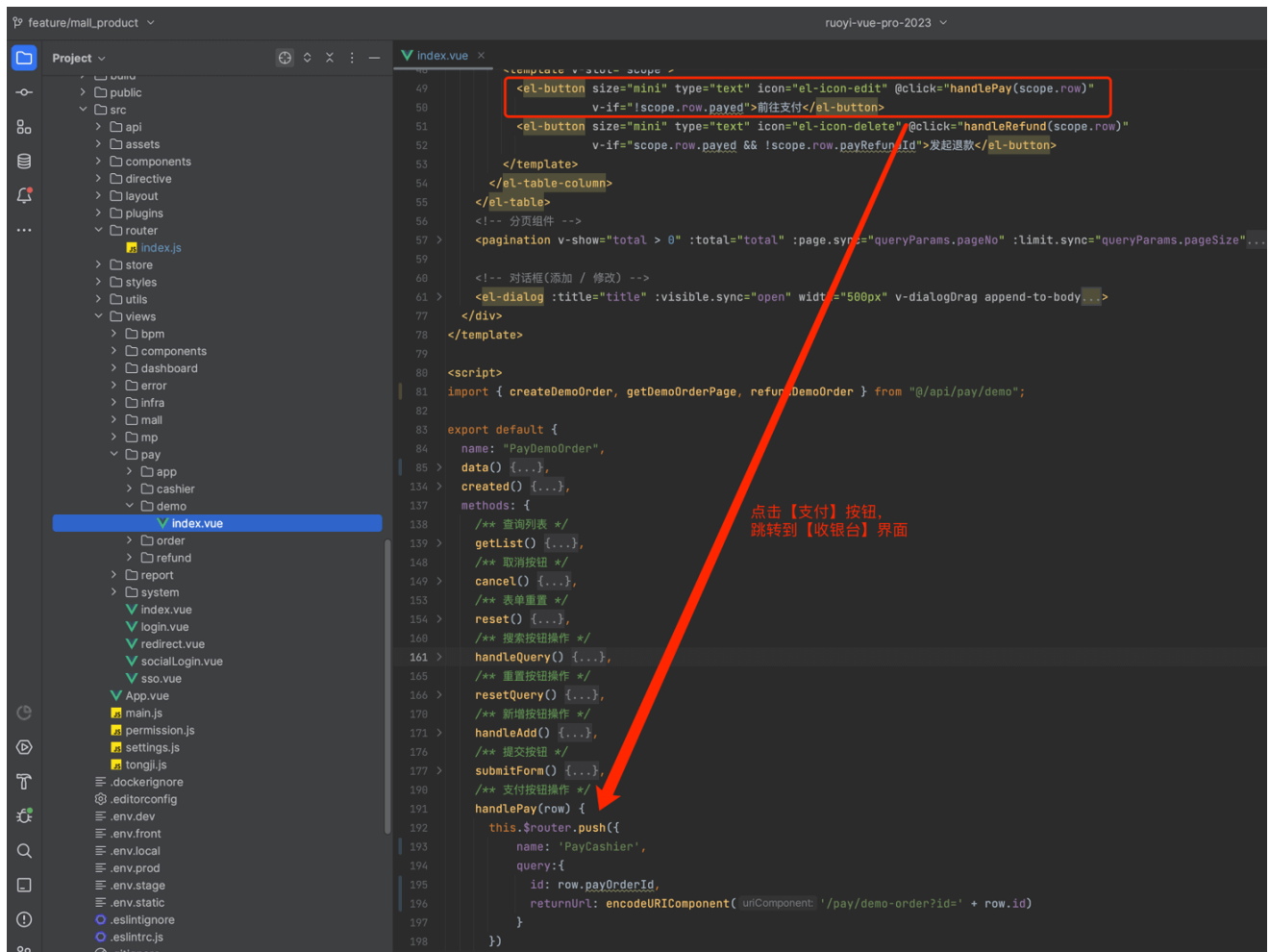


疑问：为什么 demo 模块在下单逻辑中，需要调用 PayOrderApi 接口来创建支付单？

因为跳转公用的【收银台】，需要通过读取支付订单，展示支付信息。这样，才能和 demo 模块进行解耦。收银台如下图所示：



③ 【前端】在 demo 模块下完单之后，前端需要跳转到【收银台】。前端接入代码在 `/views/pay/demo/index.vue` [🔗](#)，如下图所示：



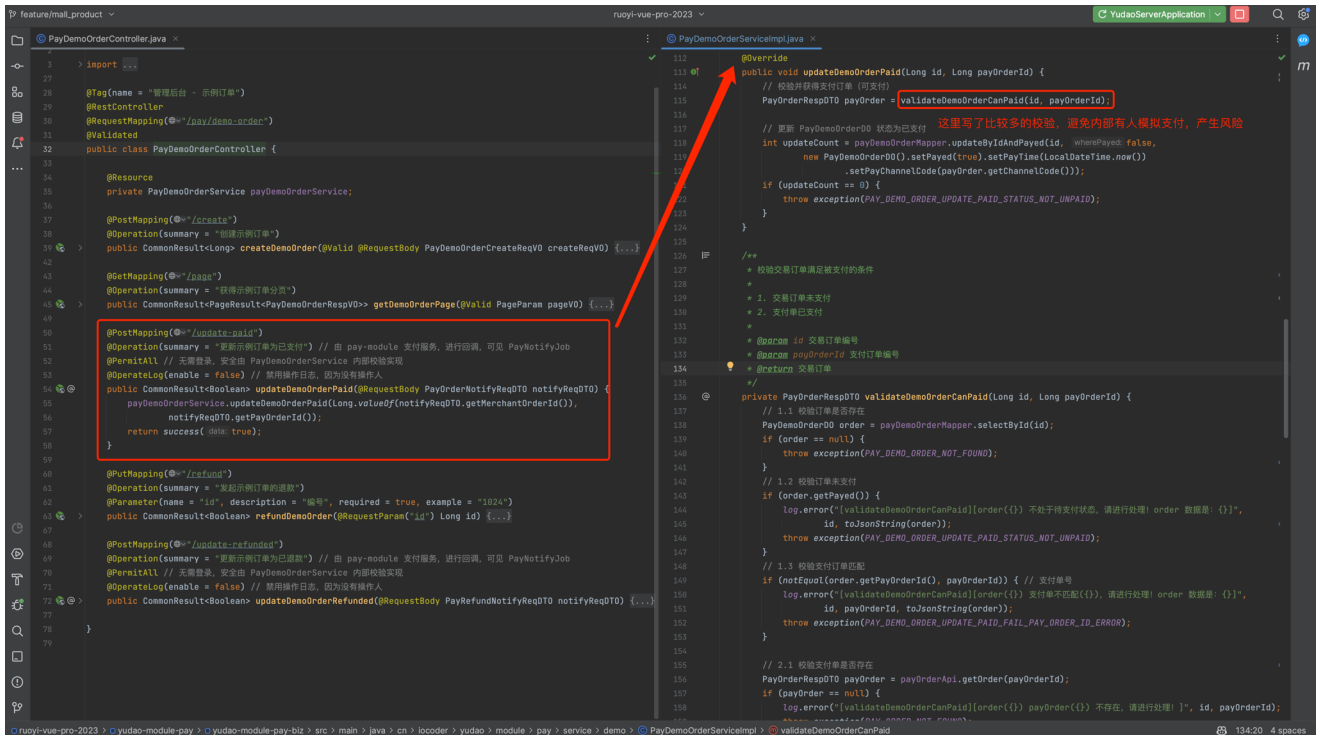
```
this.$router.push({
  name: 'PayCashier',
  query: {
    id: row.payOrderId, // 支付单号
    returnUrl: encodeURIComponent('/pay/demo-order?id=' + row.id) // 支付成功后，
  }
})
```

另外，收银台的前端代码，在 `/views/pay/cashier/index.vue` [🔗](#) 里，已经实现，感兴趣可以看看。

3. 第三步，实现回调接口【重要】

友情提示：由于 demo 模块的支付接入已经实现，这里你只要看懂什么意思即可，不用操作。

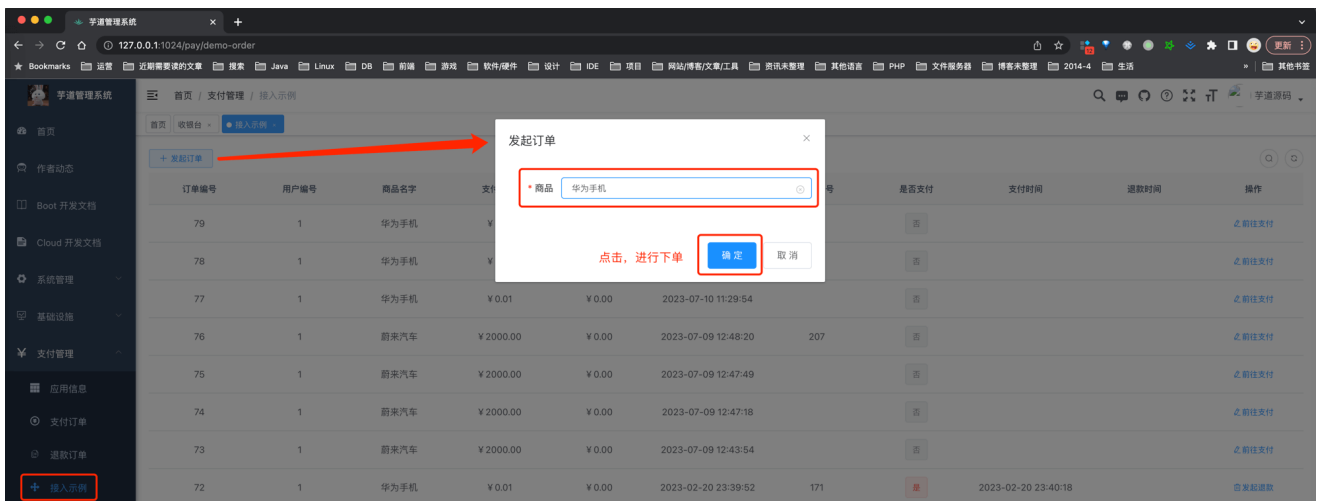
在 `demo` 模块所在的 `yudao-module-xx-biz` 模块，实现一个支付回调的接口，提供给支付【中心】回调。对应的代码在 `PayDemoOrderController` 的 `#updateDemoOrderPaid(...)` 方法中，如下图所示：



4. 第四步，支付功能测试

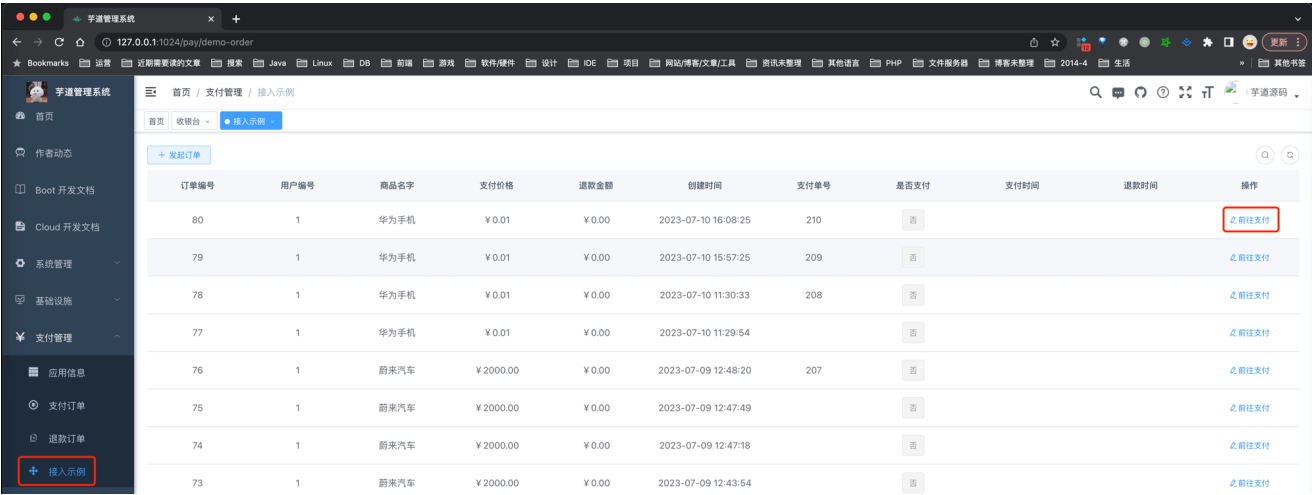
至此，我们已经完成了支付接入的所有步骤，接下来，我们来测试一下支付功能。

① 打开 [支付管理 -> 接入示例] 菜单，进入示例订单列表。点击【发起订单】按钮，选择一个商品，进行下单。如下图所示：

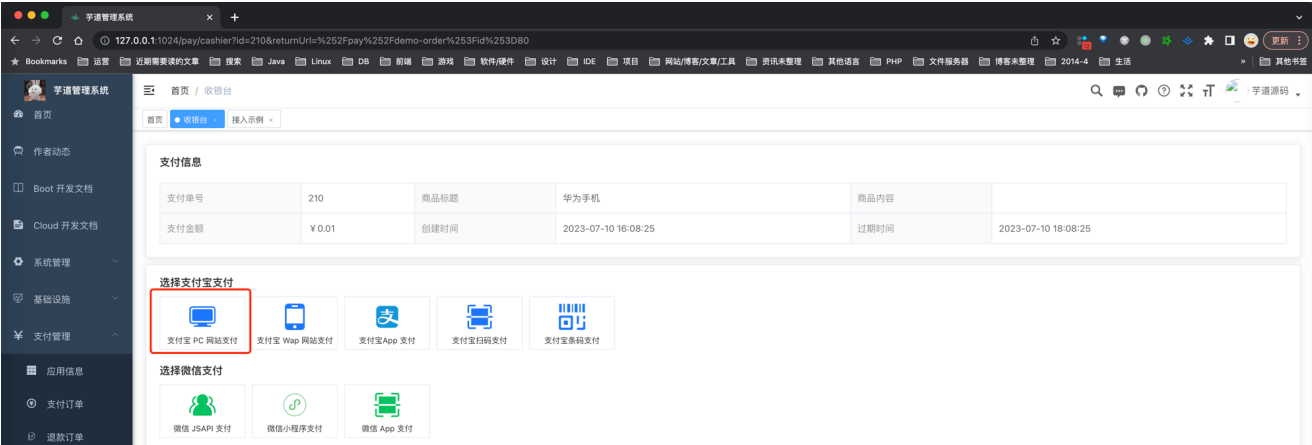


此时，在 `pay_order` 表中，会新增一条支付订单记录。

② 下单完成后，点击该订单对应的【支付】按钮，跳转到【收银台】。如下图所示：



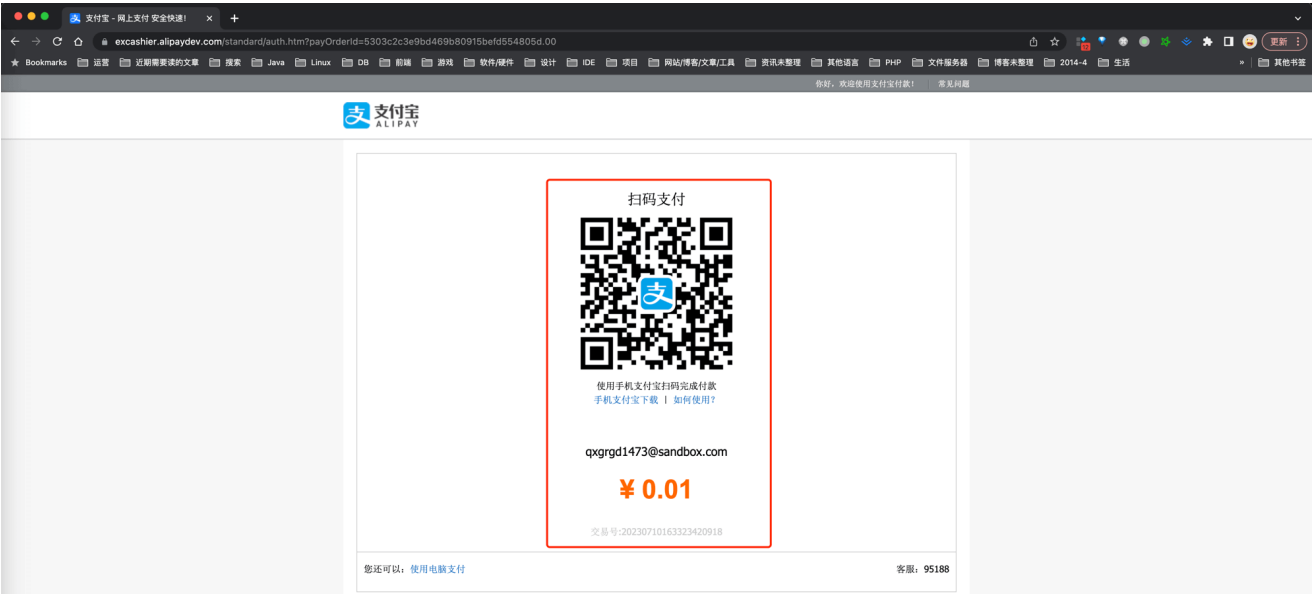
③ 选择【支付宝 PC 网站支付】支付渠道，跳转支付宝支付。如下图所示：



友情提示：

选择完支付【渠道】，会调用后端 PayOrderController 的 `#submit(...)` 方法，内部调用对应支付【渠道】的 PayClient 实现类，发起支付。

④ 此时，我们会看到一个支付宝的二维码，使用“沙箱环境”的支付宝客户端，扫码完成支付。如下图所示：



⑤ 支付完成后，先是支付【中心】的回调接口被回调，然后是 demo 模块的回调接口被回调。如下图所示：

PayNotifyController.java

```
23 @Tag(name = "管理后台 - 支付通知")
24 @RestController
25 @RequestMapping("@=/pay/notify")
26 @Validated
27 @Slf4j
28 public class PayNotifyController {
29
30     @Resource
31     private PayOrderService orderService;
32     @Resource
33     private PayRefundService refundService;
34
35     @Resource
36     private PayClientFactory payClientFactory;
37
38     @PostMapping(value = @="/order/{channelId}")
39     @Operation(summary = "支付渠道的回调-【支付】回调")
40     @PermitAll
41     @OperateLog(enable = false) // 回调地址，无需记录操作日志
42     public String notifyOrder(@PathVariable("channelId") Long channelId,
43                             @RequestBody(required = false) Map<String, String> params,
44                             @RequestParam(required = false) String body) {
45         log.info("[notifyOrder][channelId({}) 回调数据({}/{})]", channelId, params, body);
46         // 1. 接收支付渠道推送存在
47         PayClient payClient = payClientFactory.getPayClient(channelId);
48         if (payClient == null) {
49             log.error("[notifyCallback][渠道编号({}) 找不到对应的支付客户端]", channelId);
50             throw exception(PAY_CHANNEL_CLIENT_NOT_FOUND);
51         }
52
53         // 2. 解析通知数据
54         PayOrderRespDTO notify = payClient.parseOrderNotify(params, body);
55         orderService.notifyOrder(channelId, notify);
56         return "success";
57     }
58 }
```

PayDemoOrderController.java

```
18 import org.springframework.web.bind.annotation.*;
19
20 import javax.annotation.Resource;
21 import javax.annotation.security.PermitAll;
22 import javax.validation.Valid;
23
24 import static cn.iocoder.yudao.framework.common.pojo.CommonResult.success;
25 import static cn.iocoder.yudao.framework.common.util.servlet.ServletUtils.getClientIP;
26 import static cn.iocoder.yudao.framework.security.core.util.SecurityFrameworkUtils.getLoginUserId;
27
28 @Tag(name = "管理后台 - 示例订单")
29 @RestController
30 @RequestMapping("@=/pay/demo-order")
31 @Validated
32 public class PayDemoOrderController {
33
34     @Resource
35     private PayDemoOrderService payDemoOrderService;
36
37     @PostMapping("@=/create")
38     @Operation(summary = "创建示例订单")
39     public CommonResult<Long> createDemoOrder(@Valid @RequestBody PayDemoOrderCreateReqVO createReqVO) {...}
40
41     @GetMapping("@=/page")
42     @Operation(summary = "获得示例订单分页")
43     public CommonResult<PageResult<PayDemoOrderRespVO>> getDemoOrderPage(@Valid PageParam pageVO) {...}
44
45     @PostMapping("@=/update-paid")
46     @Operation(summary = "更新示例订单为已支付") // 由 pay-module 支付服务，进行回调，可见 PayNotifyJob
47     @PermitAll // 无需登录，安全由 PayDemoOrderService 内部校验实现
48     @OperateLog(enable = false) // 禁用操作日志，因为没有操作人
49     public CommonResult<Boolean> updateDemoOrderPaid(@RequestBody PayOrderNotifyReqDTO notifyReqDTO) {
50         payDemoOrderService.updateDemoOrderPaid(Long.valueOf(notifyReqDTO.getMerchantOrderId()),
51             notifyReqDTO.getPayOrderId());
52         return success(data: true); // demo 模块的回调接口，被支付【中心】回调
53     }
54 }
```

至此，我们已经完成支付接入的测试流程，可以试着多多 debug 调试整个流程，并不复杂噢。

← 功能开启

微信公众号支付接入 →