手机网站使用支付宝

手机网站是指在手机浏览器中打开的网站，在设计阶段，我们使用VUE3+Springboot开发，使用Chrome浏览器的【手机调试界面】进行调试，在使用手机浏览器可以直接输入URL访问。

开发完成后，如果需要使用APP方式在手机中安装使用，可以在android中加入webView组件，webview加载开发好的这个前端网站即可，IPHONE也类似思路，最终实现跨平台。

概念：H5是指使用移动APP中的webview作为浏览器显示的HTML页面，不同手机网站，它可以实现类似androi APP ,可以访问硬件并实现与andriod或iphone的交互.在HTML实现方面，与手机网站是一致的。

前后端分离使用支付宝沙箱实现支付功能

理解操作过程：

（1）用户单击确定支付按钮，提交支付金额到后端接口->后端接口向支付提交支付请求，获取支付宝返回数据[一个表单字符串 form]-->将结果返回给前端。

（2）用户前端收到返回数据（表单字符串），显示到页面，在显示完成后，使用JS执行（这个是关键），以表单方式，再次向支付宝提交。

（3）支付宝**自动**返回【扣款页面---支付宝页面】给用户前端，用户在支付宝扣款页面按提示执行付款操作，支付宝自动验证、自动扣款。

（4）支付宝在用户完成付款后：1）**自动**跳转到支付完成的用户设置的前端页面（后端必须配置）；2）**自动**调用设置的后端接口（后端必须配置），通知到后端已完成付款操作，后端记录订单编号、支付金额等信息。

1.创建支付宝沙箱账号：获取Appid、网关和秘钥

（1）登录支付宝 https://open.alipay.com/

（2）找到“控制台”，单击，使用支付宝扫码登录（https://open.alipay.com/develop/manage）

（3）在页面【最下方】：找到“开发工具推荐”->"沙箱"

（4）在该页面，你可以找到你所需要的 APPID、网关、秘钥、支付宝网关 等信息

（5）在该页面的“接口加签方式“-->“系统默认秘钥”-->“公钥模式”-->单击“查看”链接，找到：

```
应用私钥[注意：不是“应用公钥”]:---对应=>  privateKey
支付宝公钥:--- 对应=>   alipayPublicKey
```

在SpringBoot全局配置文件application.properties中, 填写你的这些信息

```
#登录支付宝，在控制台“沙箱应用”中找到对应值
alipay.app-id= 你的APPID
alipay.private-key=你的应用私钥
alipay.alipay-public-key=你的支付宝公钥
#固定值：沙箱使用
alipay.gateway-url=https://openapi-sandbox.dl.alipaydev.com/gateway.do

#下面两个值需要Cpolar的内网穿透功能，后面介绍如何操作[====填写公网地址 ====]
alipay.notify-url=你的后端对应的公网URL/notify接口  !!!====>》后端公网地址/接口名notify
#支付宝成功支付后，自动跳转的前端vue页面，即路由名，设置为paySuccess，你设置了才会跳转
alipay.return-url=你的前端对应的公网访问URL/paySuccess !!!===>前端公网地址/paySuccess
```

下载支付宝沙箱+查看支付宝账号

在支付宝沙箱页面==》“沙箱工具”中下载支付宝沙箱==》安装到你的手机和电脑

沙箱支付宝账号：在支付宝沙箱页面==》“沙箱账号”，查看登录到沙箱支付宝的账号和支付密码等，记住你的账号、登录密码和支付密码。

2.前端vue3

两个页面，一个路由和一个vite配置。注意安装axios组件。

2.1 前端支付页面

就一个确定支付按钮，单击该按钮，将数据提交到后端接口。

（1）首先显示“立即支付按钮”，单击该按钮，调用getAlipay函数，实现向后端接口发送wappay请求，请求参数至少包含subject、totalAmount和body三个属性，分别代表标题、总金额和订单内容描述。除了金额，其他两个内容都是自己定义的，但属性名是固定的。

（2）后端接口将返回一个支付宝生成的"form"表单字符串（该字符串在后端向支付宝发送请求获取，这里的代码在[第2部分实现]），无需理会返回的内容是什么，只要将其塞入alipayForm元素即可，然后执行该表单提交即可打开支付宝页面。

```
<template>
  <div>
    <button @click="pay">立即支付</button>
    <!-- 后端返回的表单塞到这里 -->
    <div ref="alipayForm" style="display: none"></div>
  </div>
</template>

<script setup>
import axios from 'axios'
import { nextTick, ref } from 'vue'
const alipayForm = ref(null)

//获取后端返回的表单字符串，并以HTML格式显示在alipayForm的DIV中
const pay = () => {
  getAlipay()
}

async function getAlipay() {
```

```
  // 1. 调用后端生成订单，后端返回 form 表单字符串,包含form
  await axios({
    url: 'http://localhost:8080/pay',
    method: 'post',
    data: {
      //3个必须属性
      subject: '黑马外卖3件', //显示结账的标题
      totalAmount: '0.01', //总金额
      body: '单品(1份)|套餐(2份)', //简略写商品内容
    },
  }).then((response) => {
    alipayForm.value.innerHTML = response.data //将其塞入页面元素，该值本身就是一个表单字符串

    //赋值后，等待下一个周期form生效提交，确保form生成。否则找不到form元素
    nextTick(() => {
      //自动执行表单提交
      alipayForm.value.querySelector('form').submit()
    })
  })
}
</script>
```

2.2 前端支付成功页面

该页面在支付宝付款成功后会自动调用，这里仅仅显示"支付成功",消息，以及一个返回到主页的按钮

```
<template>
  <div>
    <div>
      <button @click="back">返回商家</button>
    </div>
    <!-- <pre>{{ route.query }}</pre> -->
  </div>
</template>

<script setup>
import { useRouter } from 'vue-router'

const router = useRouter()

const back = () => {
  router.push('/pay')
}
</script>
```

```
可以通过：<pre>{{ route.query }}</pre>  来查看支付宝返回的数据：订单号，金额
```

2.3 Vue 的vite配置

为了使支付宝能够通过外网访问Vue应用（在前端付款成功后，使支付宝能够自动返回你的支付成功页面），在Vue3+版本，需要开放权限（在实际发布时，使用你自己的服务器）,如果你使用的不是vue3+，则自己搜索一下配置。

vite.config.js文件中，加入：

```
import { fileURLToPath, URL } from 'node:url'

import { defineConfig } from 'vite'
import vue from '@vitejs/plugin-vue'
import vueDevTools from 'vite-plugin-vue-devtools'

// https://vite.dev/config/
export default defineConfig({
    //仅仅加入该配置=====================================
  server: {
    allowedHosts: ['.cpolar.cn'], // 允许所有CPLOAR主机访问
  },

  plugins: [vue(), vueDevTools()],
  resolve: {
    alias: {
      '@': fileURLToPath(new URL('./src', import.meta.url)),
    },
  },
})
```

2.4 路由

```
import { createRouter, createWebHistory } from 'vue-router'

const router = createRouter({
  history: createWebHistory(import.meta.env.BASE_URL),
  routes: [
    {
      path: '/',
      redirect: '/pay',
    },
    {
      path: '/pay',
      name: 'pay',
      component: () => import('@/views/Pay.vue'),
    },
    {
      path: '/paySuccess', //注意：对应springboot配置中的alipay配置的return-url的属性
      name: 'paySuccess',
      component: () => import('@/views/PaySuccess.vue'),
    },
  ],
})

export default router
```

3.后端SpringBoot3

3.1 步骤

1.加入支付宝的maven坐标。

2.构建支付宝的请求参数

3.创建支付宝请求对象，发送请求，获取返回的表单数据

4.在前端访问“立即支付”的接口时，调用步骤3创建的接口，将支付宝接口返回的表单字符串数据发送给前端（前端这是自动提交表单：1.1节）

5.支付宝在成功完成用户支付后，调用本系统的后端接口（该接口在步骤2中填写），告知后端用户完成支付，将用户信息保存到订单数据库

完成。

3.2 具体实现

（1）在POM.xml文件中加入支付宝maven坐标

```
<!--         支付宝沙箱 |  支付宝 SDK  |  2024-08 官方最新 -->
        <dependency>
            <groupId>com.alipay.sdk</groupId>
            <artifactId>alipay-sdk-java</artifactId>
            <version>4.39.70.ALL</version>
        </dependency>

        <!-- 配置提示（可选） -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-configuration-processor</artifactId>
            <optional>true</optional>
        </dependency>
```

（2）构建支付宝接口请求参数

（1）新建一个访问支付宝接口必须的参数类AlipayProperties---全局属性配置类，添加支付宝接口所要求的参数属性

```
@Data
@Component
@ConfigurationProperties(prefix = "alipay")
public class AlipayProperties {
    String appId;//支付宝应用程序appId
    String gatewayUrl;//支付宝沙箱网关，固定值
    String privateKey;//支付宝沙箱私钥
    String alipayPublicKey;//支付宝公钥
    String notifyUrl;//通知后端接口URL  ---在后台记录支付成功后的信息
    String returnUrl;//前端页面URL---在支付成功后，支付宝自动返回的前端页面URL
}
```

注意：前4个属性，在支付宝沙箱应用控制台查看。notifyUrl是后端系统的访问接口，必须具有公网访问能力（这个借助Cpolar实现内网穿透。第3部分介绍），returnUrl是Vue前端页面URL，在1.2中介绍了，这个同样需要内网穿透；后两个接口如果部署到云服务器，则不需要内网穿透（该工具Cpolar用于本地测试）。

（2）在全局配置文件application.properties填写属性值， 这些属性将自动注入到PayConfig类

在创建好配置类PayConfig后，重新编译你的系统，这样在写下面属性会出现语法补全功能

```
#登录支付宝，在控制台“沙箱应用”中找到对应值
alipay.app-id=你的应用程序app-id
alipay.private-key=你的私钥
alipay.alipay-public-key=你的支付宝公钥
#固定值：沙箱使用
alipay.gateway-url=https://openapi-sandbox.dl.alipaydev.com/gateway.do

#下面两个值需要Cpolar的内网穿透【cploar生成的URL+前端路由名，或后端接口名】
alipay.notify-url=你的后端对应的公网URL/notify
#支付宝成功支付后，自动跳转的前端vue页面，你设置了才会跳转
alipay.return-url=你的前端对应的公网访问URL/paySuccess
```

（3）添加配置类，创建支付宝付款请求对象，准备使用该对象调用支付宝接口

创建一个配置类，添加一个方法，返回支付宝沙箱付款请求对象。

该对象将注入到服务类使用

```
package com.dish.alipay.config;

import com.alipay.api.AlipayClient;
import com.alipay.api.DefaultAlipayClient;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
public class AlipayConfig {

    //注入属性类
    @Autowired
    AlipayProperties props;

    @Bean
    //创建AlipayClient接口对象，通过创建DefaultAlipayClient对象返回
    public AlipayClient alipayClient() {
        AlipayClient client = new DefaultAlipayClient(
                props.getGatewayUrl(),//沙箱网关
                props.getAppId(),//应用app-id
                props.getPrivateKey(),//私钥
                "json",//固定
                "utf-8",//固定
                props.getAlipayPublicKey(),//支付宝公钥
                "RSA2"//固定
        );
        return client;
    }
```

```
}
```

（4）服务类：使用支付宝付款接口

创建服务类，调用接口，实现发送请求到支付宝

（1）注入AlipayClient对象；注入配置属性

（2）创建手机页面请求对象request，通过配置属性，设置请求对象request的属性：跳转URL；

（3）创建并设置request 对象的数据对象，该数据对象包含前端发送过来的订单信息：subject\totalAmount\body

（4）调用请求方法，实现向支付发起付款请求

（5）支付宝将返回一个表单字符串，该字符串稍后返回给前端

```
package com.dish.alipay.service;

import com.alipay.api.AlipayApiException;
import com.alipay.api.AlipayClient;
import com.alipay.api.domain.AlipayTradePayModel;
import com.alipay.api.internal.util.AlipaySignature;
import com.alipay.api.request.AlipayTradePagePayRequest;
import com.alipay.api.request.AlipayTradeWapPayRequest;
import com.alipay.api.response.AlipayTradePagePayResponse;
import com.alipay.api.response.AlipayTradeWapPayResponse;

import com.dish.alipay.config.AlipayProperties;
import com.dish.alipay.pojo.PayParam;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.Map;

@Service
public class AlipayService {
    @Autowired
    AlipayClient alipayClient;
    @Autowired
    AlipayProperties props;

    //以规定的格式（专门对象：model，将数据）包装好，发送(专门对象 request)给支付宝
    //支付收到数据后，生成一个form 数据，让前端执行提交（付款）
    //前端提交后，支付宝执行完毕，通过你提供的“notify”后端接口，发送给后端
    //后端接收到消息，如何告知前端？--->后端设置数据标记（你可以查询数据库是否完成订单标记---同步保存在redis中，是想快速查询）
    //前端会自动跳转到成功支付的自定义页面


    public String preWapCreateOrder(PayParam payParam) throws AlipayApiException {

        //构建向支付发送请求和回应的URL的请求对象，注意是trandeWap不是trandePage
        AlipayTradeWapPayRequest request = new AlipayTradeWapPayRequest();
        //设置前后端跳转URL
        request.setReturnUrl(props.getReturnUrl());
```

```
        request.setNotifyUrl(props.getNotifyUrl());

        //构建支付数据对象
        AlipayTradePayModel model = new AlipayTradePayModel();

        //自定义订单编号，这里是:ORDER_当前时间的秒数
        model.setOutTradeNo("ORDER_" + System.currentTimeMillis());
        //以下必填
        model.setSubject(payParam.getSubject());
        model.setTotalAmount(payParam.getTotalAmount());
        model.setBody(payParam.getBody());

        //针对手机网站=QUICK_WAP_PAY是固定值  在手机浏览器打开的网站
        model.setProductCode("QUICK_WAP_PAY");

        //通过setBizModel方法附件数据，将数据赋值给请求对象
        request.setBizModel(model);

        //执行支付请求，返回响应对象##################################################
        AlipayTradeWapPayResponse repsonse = alipayClient.pageExecute(request);

        //响应体为生成的表单字符串：<form></form>
        //返回给前端，前端接收到该数据，执行表单提交操作，这样，前端就完成了真正的发送了支付操
作
        return repsonse.getBody(); //该数据将在用户前端访问时返回给前端
    }


    /**
     * 异步通知验签：支付宝执行付款后，会调用该接口，返回付款信息，这里验证信息是否付款成功
     * 如果返回值是 true ，则可以将用户订单信息保存到数据库*****************************
     * params参数包含支付的所有信息：subject\body\totalAmount以及订单编号
out_trade_no、用户和商家支付宝ID等
     */
    public boolean verify(Map<String, String> params) throws Exception {
        return AlipaySignature.rsaCheckV1(params,
                props.getAlipayPublicKey(),
                "UTF-8",
                "RSA2");
    }
}
```

PayParam参数是自定义类，在控制器中接收前端（1.1中页面）提交的数据

```
package com.dish.alipay.pojo;
import lombok.Data;

@Data
public class PayParam {
    String subject;//支付标题：在支付宝付款页面出现
    String totalAmount;//付款金额
    String body;//附件你所需要的信息，比如商品名称、数量等
}
```

(5) 控制器：用户确认付款

接收用户提交的订单信息，调用支付宝付款接口，返回支付宝生成的表单。

```
 package com.example.alipay;


import com.alipay.api.AlipayApiException;
import jakarta.servlet.http.HttpServletRequest;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import java.util.HashMap;
import java.util.Map;

@RestController
public class AlipayController {
    @Autowired
    AlipayService service;


    //手机网站
    @PostMapping("/pay")
    public String preCreateOrder(@RequestBody PayParam payParam) throws
AlipayApiException {
        System.out.println(payParam);
        return service.preCreateOrder(payParam);
    }

    /**
     * 异步通知
     */
    @PostMapping("/notify")
    public String notify(HttpServletRequest request) throws Exception {
        Map<String, String[]> parameterMap = request.getParameterMap();
        Map<String, String> params = new HashMap<>();
        parameterMap.forEach( (k, v) -> params.put(k, v.length > 0 ? v[0] :
""));

        System.out.println(">>>>");

        if (service.verify(params)) {
            // TODO: 业务处理（验签成功）
            //从参数out_trade_no=ORDER_1755131710159 中更新订单状态
            System.out.println("notify success: " + params);
            return "success";
        }
        return "fail";
    }
}
```

preCreateOrder返回的字符串表单格式示例：

```
 * <form name="punchout_form" method="post" action="https://openapi-
sandbox.dl.alipaydev.com/gateway.do?charset=utf-
8&method=alipay.trade.page.pay&sign=
    * return_url=http%3A%2F%2F127.0.0.1%3A5173%2FpaySuccess&
    * notify_url=http%3A%2F%2F175.178.250.201%2Fnotify&
    * version=1.0&app_id=9021000151617278&
    * sign_type=... ">
<input type="hidden" name="biz_content" value="{&amp;quot;body&amp;quot;:&amp;quot;BODY数据
&amp;quot;,&amp;quot;out_trade_no&amp;quot;:&amp;quot;ORDER_1755076461850&amp;quot;,&amp;quot;product_co
de&amp;quot;:&amp;quot;&amp;quot;,&amp;quot;subject&amp;quot;:&amp;quot;测试
&amp;quot;,&amp;quot;total_amount&amp;quot;:&amp;quot;0.01&amp;quot;}">
<input type="submit" value="立即支付" style="display:none" >
</form>
<script>document.forms[0].submit();</script>
```

在前端使用：document.write(上面数据)；将自动立即提交表单，跳转到支付宝页面，在本文的1.1中是手动提交表单。

付款成后，返回的对象信息params格式示例：

```
/ * * {
    * gmt_create=2025-08-14 08:35:23,
    * charset=utf-8,
    * seller_email=nfugpj2103@sandbox.com,
    * subject=测试商品，  ---这里可以写商家名称
    * body=沙箱支付测试，  ---
    * buyer_id=2088722075858484, //用户ID!===>支付宝用户ID用户 UID2088722075858484
    * invoice_amount=0.01,
    * notify_id=2025081401222083525158480507062366,
    * fund_bill_list=[{"amount":"0.01","fundChannel":"ALIPAYACCOUNT"}],//付款方式
    * notify_type=trade_status_sync,
    * trade_status=TRADE_SUCCESS,---付款状态
    * receipt_amount=0.01,  ---收到金额
    * buyer_pay_amount=0.01, ---付款金额
    * app_id=9021000151617278,
    * seller_id=2088721075858472, //商户ID
    * gmt_payment=2025-08-14 08:35:24, ---支付时间
    * notify_time=2025-08-14 08:35:26,---付款成功通知的时间
    * version=1.0,
    * //订单信息
    * out_trade_no=ORDER_1755131710159,  ----订单号
    * total_amount=0.01, ---付款金额
    * trade_no=2025081422001458480506955150,  //商户号
    * auth_app_id=9021000151617278,
    *  buyer_logon_id=svxyuf9482@sandbox.com, ----用户支付宝账号
    *  point_amount=0.00 ---返点金额
    * }
*/
```

（6）跨越配置

```
package com.example.alipay;

import org.springframework.context.annotation.Configuration;
```

```
import org.springframework.web.servlet.config.annotation.CorsRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

@Configuration
public class WebMvcConfig implements WebMvcConfigurer {


//    你自己选择跨域方式的实现
    @Override
    public void addCorsMappings(CorsRegistry registry) {
//        WebMvcConfigurer.super.addCorsMappings(registry);

        registry.addMapping("/**")
                .allowedOrigins("*")
                .allowedMethods("*")
                .allowedHeaders("*");
    }
}
```

3.3 支付问题

手机端网站，点击支付宝付款链接，无法激活（唤起）支付宝沙箱，只能使用【选项2：登录支付

】能够完全模拟实现支付

4.Cpolar入门

一个内网穿透工具，简单理解为：可以将你本地服务器，转换为可以全球访问的公网服务器。

在本地调试支付宝沙箱功能必备工具。【你也可以使用花生壳】

4.1 使用Cpolar-构建notifyUrl和returnUrl请求参数

目的：将本地URL，映射为外网可以访问的URL，这样支付宝在付款后，可以自动跳转【这样支付宝才能与你应用交互】到前端页面和付款后通知后端付款成功与否的状态信息。

4.2 步骤

（1）注册Cpolar账号，下载和安装Cpolar客户端（免费）https://www.cpolar.com/

（2）安装好，在浏览器地址栏输入(cpolar默认URL)：http://localhost:9200打开Cpolar操作界面，使用注册的账号登录。

（3）在控制台创建隧道，从而获取公网URL

4.3 具体操作

（1）打开浏览器，输入http://localhost:9200 ，打开Cpolar界面

（2）在“隧道管理”->"创建隧道"，创建两个隧道：

```
前端Vue ，假如本地默认地址URL：http://localhost:5173
那么：
1.在“隧道名称”栏，随意输入一个名字，例如：前端（Vue）服务器
2.在“协议”栏,选择http
3.在"本地地址"栏：输入：http://localhost:5173


后端服务器，假如本地访问地址为：http://localhost:8080
那么：
1.在“隧道名称”栏，随意输入一个名字，例如：后端服务器
2.在“协议”栏,选择http
3.在"本地地址"栏：输入：http://localhost:8080
```

（3）查看创建的隧道

在“隧道列表”中，可以看到当前创建的隧道，如果处于未启动状态，那么单击“启动”按钮

（4）使用当前内网穿透URL

在"状态"->“在线隧道列表”中，复制你的前后端对应的“公网地址”，http或者https均可。

（5）在SpringBoot全局配置文件application.properties中, 填写你找到的公网地址

```
#登录支付宝，在控制台“沙箱应用”中找到对应值
alipay.app-id=你的应用程序app-id
alipay.private-key=你的私钥
alipay.alipay-public-key=你的支付宝公钥
#固定值：沙箱使用
alipay.gateway-url=https://openapi-sandbox.dl.alipaydev.com/gateway.do

#下面两个值需要Cpolar的内网穿透能录[====填写获取的公网URL ======================]
alipay.notify-url=你的后端对应的公网URL/notify  !!!====》》》》》后端公网地址/接口名
#支付宝成功支付后，自动跳转的前端vue页面[Vue路由中的名字为：paySuccess]，你设置了才会跳转
alipay.return-url=你的前端对应的公网访问URL/paySuccess !!!====》》》》》前端公网地
址/paySuccess
```

例如我的配置是：

```
#每次启动cploar都随机的，不一样
alipay.return-url=http://251e2640.r3.cpolar.cn/paySuccess
alipay.notify-url=http://4f64a89c.r3.cpolar.cn/notify
```

小技巧：借助二维码生成工具，将http://251e2640.r3.cpolar.cn生成二维码，在手机浏览器中，可以扫码访问你的前端app，这样你就不需要自己手工录入前端访问的URL来测试了。

```
https://www.67tool.com/work/qrcode?identity=53d466
```

5.电脑网站（PC端网站）实现支付宝沙箱功能

与手机网站实现的不同之处：1.使用请求对象不同；2.产品代码属性字符串不同。其它都是一样的。

以下代码是在使用**PC端浏览器**打开你的前端（cpolar提供的访问URL）

```
    //以规定的格式（专门对象：model，将数据）包装好，发送(专门对象 request)给支付宝
    //支付收到数据后，生成一个form 数据，让前端执行提交（付款）
    //前端提交后，支付宝执行完毕，通过你提供的“notify”后端接口，发送给后端
    //后端接收到消息，如何告知前端？--->后端设置数据标记（你可以查询数据库是否完成订单标记---同步保存在redis中，是想快速查询）
    //前端会自动跳转到成功支付的自定义页面
    public String preCreateOrder(PayParam payParam) throws AlipayApiException {
        System.out.println(props);

        //不同之处1：构建向支付发送请求和回应的URL的请求对象================
        AlipayTradePagePayRequest request = new AlipayTradePagePayRequest();

        //手机网站
        // AlipayTradeWapPayRequest request = new AlipayTradeWapPayRequest();


        request.setReturnUrl(props.getReturnUrl());
        request.setNotifyUrl(props.getNotifyUrl());

        //构建支付数据
        AlipayTradePayModel model = new AlipayTradePayModel();
        model.setOutTradeNo("ORDER_" + System.currentTimeMillis());
        //以下必填
        model.setSubject(payParam.getSubject());
        model.setTotalAmount(payParam.getTotalAmount());
        model.setBody(payParam.getBody());

        //网页支付，固定的字符串

        //不同之处2：电脑网站：============================
        model.setProductCode("FAST_INSTANT_TRADE_PAY");


         //如果手机网站：在手机浏览器打开的网站
         //model.setProductCode("QUICK_WAP_PAY");


        //将数据赋值给请求对象
        request.setBizModel(model);
        //执行支付
        AlipayTradePagePayResponse repsonse = alipayClient.pageExecute(request);

        //响应体为生成的表单字符串：<form></form>
        //返回给前端，前端接收到该数据，执行表单提交操作，这样，前端就完成了真正的发送了支付操作
        return repsonse.getBody();
    }
```

前端不变。