# 二维码的使用

二维码介绍

二维码又称QR Code，QR全称Quick Response，是一个近几年来移动设备上超流行的一种编码方式，它比传统的Bar Code条形码能存更多的信息，也能表示更多的数据类型。

二维条码/二维码（2-dimensional bar code）是用某种特定的几何图形按一定规律在平面（二维方向上）分布的黑白相间的图形记录数据符号信息的；在代码编制上巧妙地利用构成计算机内部逻辑基础的“0”、“1”比特流的概念，使用若干个与二进制相对应的几何形体来表示文字数值信息，通过图象输入设备或光电扫描设备自动识读以实现信息自动处理：它具有条码技术的一些共性：每种码制有其特定的字符集；每个字符占有一定的宽度；具有一定的校验功能等。同时还具有对不同行的信息自动识别功能、及处理图形旋转变化点。

二维码优势

 信息容量大, 可以容纳多达1850个大写字母或2710个数字或500多个汉字

 应用范围广, 支持文字,声音,图片,指纹等等...

 容错能力强, 即使图片出现部分破损也能使用

 成本低, 容易制作

二维码容错级别

L级（低） 7％的码字可以被恢复。

M级（中） 的码字的15％可以被恢复。

Q级（四分）的码字的25％可以被恢复。

H级（高） 的码字的30％可以被恢复。

插件qrious生成二维码步骤

rious.js二维码插件的可用配置参数如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 默认值 | 描述 |
| background | String | "white" | 二维码的背景颜色。 |
| foreground | String | "black" | 二维码的前景颜色。 |
| level | String | "L" | 二维码的误差校正级别(L, M, Q, H)。 |
| mime | String | "image/png" | 二维码输出为图片时的MIME类型。 |
| size | Number | 100 | 二维码的尺寸，单位像素。 |
| value | String | "" | 需要编码为二维码的值 |

不需要导入jquer插件，只需要qrious.min.js即可

<html>

<head>

<title>二维码入门小demo</title>

</head>

<body>

<img id="qrious"> <!-- 第一步 需要使用标签img -->

<script src="qrious.min.js"></script>

<script>

<!-- 第二步 使用方法new Qious 参数是js对象 -->

var qr = new QRious({

element:document.getElementById('qrious'), //获取img的id标签

size:250, //制定二维码大小

level:'H', //二维码的误差校正级别

value:'http://www.baidu.cn' //二维码的值

});

</script>

这样两步：就能在页面生成二维码

扫码支付的步骤

大致原理

其实就是，通过一系列的操作后，微信给你发来url地址支付页面

客户通过扫码进入支付页面：进行支付即可

页面的内容需要自己实现：如价格，支付完成后的(还需要调微信的接口来判断是否支付成功)跳转页面

第一步 使用httpclient调微信支付接口：返回微信的支付url地址（用户扫码后跳转支付地址）

public Map createNative(String out\_trade\_no, String total\_fee) {//交易订单号与支付金额

//1.参数封装：微信必须需要的参数封装

Map param=new HashMap();

param.put("appid", appid);//公众账号ID，appid从配置文件中读取

param.put("mch\_id", partner);//商户 从配置文件中读取

param.put("nonce\_str", WXPayUtil.generateNonceStr());//随机字符串

param.put("body", "品优购");//调用支付接口，用户需要支付的提示

param.put("out\_trade\_no", out\_trade\_no);//交易订单号

param.put("total\_fee", total\_fee);//金额（分）

param.put("spbill\_create\_ip", "127.0.0.1");

param.put("notify\_url", "http://www.itcast.cn");//回调页面，因使用第二种方式：这里按格式写

param.put("trade\_type", "NATIVE");//交易类型

try {

String xmlParam = WXPayUtil.generateSignedXml(param, partnerkey);

System.out.println("请求的参数："+xmlParam);

//2.发送请求

HttpClient httpClient=new HttpClient("https://api.mch.weixin.qq.com/pay/unifiedorder");

httpClient.setHttps(true);

httpClient.setXmlParam(xmlParam);

httpClient.post();

//3.获取结果

String xmlResult = httpClient.getContent();

//最终返回来的结果，主要需要支付的url

Map<String, String> mapResult = WXPayUtil.xmlToMap(xmlResult);

System.out.println("微信返回结果"+mapResult);

//不需要敏感的数据(公众号ID等) 进行再次封装，这里只取出支付url地址

Map map=new HashMap<>();

map.put("code\_url", mapResult.get("code\_url"));//生成支付二维码的链接url

map.put("out\_trade\_no", out\_trade\_no);

map.put("total\_fee", total\_fee);

return map;

} catch (Exception e) {

// TODO Auto-generated catch block

e.printStackTrace();

return new HashMap();

}

}

第二步：判断用户是否支付成功

需要调微信的另一个接口

需要在控制层进行while循环来调用service层调用微信接口判断是否成功

public Map queryPayStatus(String out\_trade\_no) {

Map param=new HashMap();

param.put("appid", appid);//公众账号ID

param.put("mch\_id", partner);//商户号

param.put("out\_trade\_no", out\_trade\_no);//订单号

param.put("nonce\_str", WXPayUtil.generateNonceStr());//随机字符串

String url="https://api.mch.weixin.qq.com/pay/orderquery";

try {

String xmlParam = WXPayUtil.generateSignedXml(param, partnerkey);

HttpClient client=new HttpClient(url);

client.setHttps(true);

client.setXmlParam(xmlParam);

client.post();

String result = client.getContent();

Map<String, String> map = WXPayUtil.xmlToMap(result);

System.out.println(map);

return map;

} catch (Exception e) {

e.printStackTrace();

return null;

}

}

控制层

@RequestMapping("/queryPayStatus")

public Result queryPayStatus(String out\_trade\_no){

Result result=null;

**int** x=0;

while(true){

//调用查询接口

Map<String,String> map = weixinPayService.queryPayStatus(out\_trade\_no);

if(map==null){//出错

result=new Result(false, "支付出错");

break;

}

if(map.get("trade\_state").equals("SUCCESS")){//如果支付成功微信返回trade\_state: SUCCESS

result=new Result(true, "支付成功");

break;

}

try {

Thread.sleep(3000);//间隔三秒

} catch (InterruptedException e) {

e.printStackTrace();

}

//为了不让循环无休止地运行，我们定义一个循环变量，如果这个变量超过了这个值则退出循环，设置时间为5分钟

x++;

**if**(x>=100){ //上面睡觉3秒一次循环循环100次相当于5分钟

result=**new** Result(**false**, "二维码超时");

**break**;

}

}

return result;

}

前端的操作：获取到数据进行判断：

# 正则表达式

## test()使用方式

操作事项：

1使用的是纯js的test()方法；

2 new RegExp方法只是检查正则表达式是否正确

3 用正则表达式的变量来调用test()方法：符合正则返回true

//第一步：设置正则表达式变量

var re=/^\w+([-\.]\w+)\*@\w+([\.-]\w+)\*\.\w{2,4}$/; //没有双引号或单引号

//第二步：验证正则表达式是否正确(可省略此步骤)

var szemailReg = new RegExp(re);

//第三步:使用正则表达式的变量进行调用test方法：test(”这里是要验证的内容”)

if(szemailReg.test(szemail)){

return true;

}else{

return false;

}

表达式

邮箱：var re=/^\w+([-\.]\w+)\*@\w+([\.-]\w+)\*\.\w{2,4}$/

手机号码: var szemailReg=/^\w+([-\.]\w+)\*@\w+([\.-]\w+)\*\.\w{2,4}$/;

身份证验证：var arg1 = /^[1-9]\d{7}((0\d)|(1[0-2]))(([0|1|2]\d)|3[0-1])\d{3}$/;

网上内容：https://www.cnblogs.com/chenmfly/p/5285451.html