**品优购电商系统开发**

**第18章**

**提交订单与微信扫码支付**

传智播客.黑马程序员

# 1.结算页-收件人地址选择

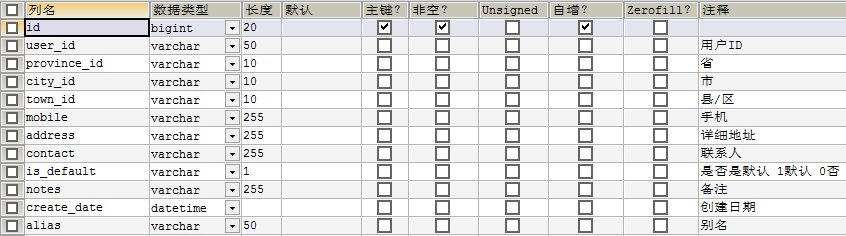
## 1.1需求与数据库分析

### 1.1.1需求描述

在结算页实现收件人地址选择功能

### 1.1.2数据库结构分析

Tb\_address 为地址表



## 1.2准备工作

### 1.2.1后端代码

1. interface项目编写接口AddressService
2. user-service 项目编写实现类AddressServiceImpl
3. portal-web项目编写AddressController

### 1.2.2拷贝页面资源

将getOrderInfo.html拷贝至portal-web的webapp下

## 1.3实现地址列表

### 1.3.1后端代码

修改user-service的AddressServiceImpl.java

|  |
| --- |
| /\*\*  \* 收货地址管理  \* **@author** lx  \*  \*/  @Service  **public** **class** AddressServiceImpl **implements** AddressService {    @Autowired  **private** AddressDao addressDao;  //通过用户ID查询收货地址结果集  **public** List<Address> findListByLoginUser(String name){  AddressQuery addressQuery = **new** AddressQuery();  addressQuery.createCriteria().andUserIdEqualTo(name);  **return** addressDao.selectByExample(addressQuery);  }  } |

1. 修改portal-web的AddressController.java

|  |
| --- |
| /\*\*  \* 收货地址  \* **@author** lx  \*  \*/  @RestController  @RequestMapping("/address")  **public** **class** AddressController {  @Reference  **private** AddressService addressService;  //通过用户ID查询收货地址结果集  @RequestMapping("/findListByLoginUser")  **public** List<Address> findListByLoginUser(){  String name = SecurityContextHolder.*getContext*().getAuthentication().getName();    **return** addressService.findListByLoginUser(name);  }  } |

### 1.3.2前端代码

（1）portal-web的cartService.js

|  |
| --- |
| //获取地址列表  **this**.findAddressList=**function**(){  **return** $http.get('address/findListByLoginUser.do');  } |

（2）portal-web的cartController.js

|  |
| --- |
| //获取地址列表  $scope.findAddressList=**function**(){  cartService.findAddressList().success(  **function**(response){  $scope.addressList=response;  }  );  } |

（3）修改getOrderInfo.html

引入 js

|  |
| --- |
| <script type=*"text/javascript"* src=*"plugins/angularjs/angular.min.js"*> </script>  <script type=*"text/javascript"* src=*"js/base.js"*> </script>  <script type=*"text/javascript"* src=*"js/service/cartService.js"*> </script>  <script type=*"text/javascript"* src=*"js/controller/cartController.js"*> </script> |

指令

|  |
| --- |
| <body ng-app=*"pinyougou"* ng-controller=*"cartController"* ng-init=*"findAddressList()"*> |

循环列表

|  |
| --- |
| <div ng-repeat=*"address in addressList"* >  <div class=*"con name"*>  <a href=*"javascript:;"* >{{address.contact}}<span title=*"点击取消选择"*>&nbsp;</a>  </div>  <div class=*"con address"* >{{address.address}}<span>{{address.mobile}}</span>  <span class=*"base"* ng-if=*"address.isDefault=='1'"*>默认地址</span>  <span class=*"edittext"*><a data-toggle=*"modal"* data-target=*".edit"* data-keyboard=*"false"* >编辑</a>&nbsp;&nbsp;<a href=*"javascript:;"*>删除</a></span>  </div>  <div class=*"clearfix"*></div>  </div> |

显示效果如下：



# 2.保存订单

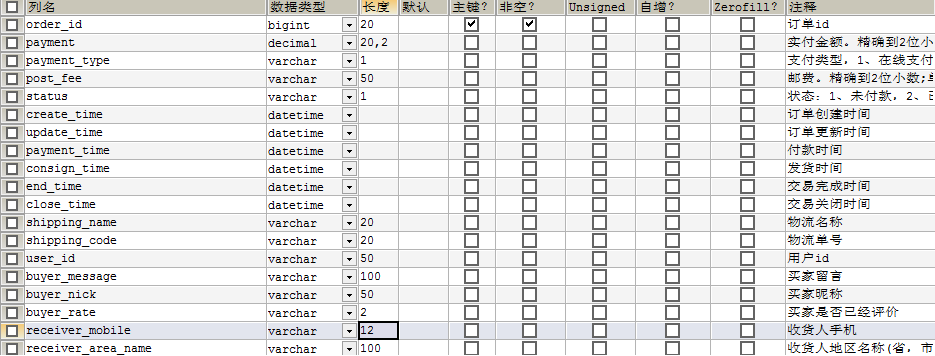
## 2.1需求分析

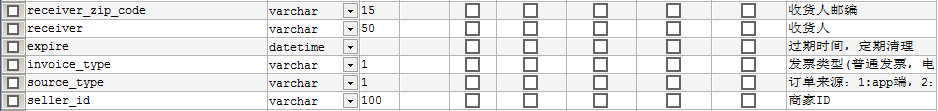
### 2.1.1需求描述

点击订单结算页的提交订单 ，将购物车保存到订单表和订单明细表中，并将购物车数据清除.

### 2.1.2数据库结构分析

Tb\_order为订单主表





## 2.2准备工作

### 2.2.1编写后端代码

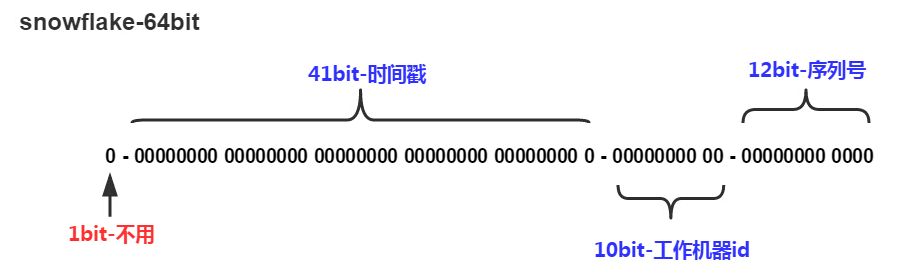
interface项目编写OrderService.java

buyer-service项目编写OrderServiceImpl.java

portal-web项目编写OrderController.java

### 2.2.2分布式ID生成器

我们采用的是开源的twitter( 非官方中文惯称：推特.是国外的一个网站，是一个社交网络及微博客服务) 的snowflake算法。



（1）将“配置文件/工具类”下的IdWorker.java 拷贝到common工程

（2）在buyer-service工程的spring配置文件中添加配置

|  |
| --- |
| <bean id=*"idWorker"* class=*"util.IdWorker"*>  <!-- 进程ID -->  <constructor-arg index=*"0"* value=*"0"*></constructor-arg>  <!-- 数据中心ID -->  <constructor-arg index=*"1"* value=*"0"*></constructor-arg>  </bean> |

## 2.3后端代码

### 2.3.1服务实现层

修改buyer-service的OrderServiceImpl.java

|  |
| --- |
| /\*\*  \* 保存订单  \* **@author** lx  \*  \*/  @Service  @Transactional  **public** **class** OrderServiceImpl **implements** OrderService {    @Autowired  **private** OrderDao orderDao;  @Autowired  **private** OrderItemDao orderItemDao;  @Autowired  **private** RedisTemplate redisTemplate;  @Autowired  **private** CartService cartService;  @Autowired  **private** PayLogDao payLogDao;    @Autowired  **private** IdWorker idWorker;  //保存订单  **public** **void** add(Order order){  List<Cart> cartList = cartService.  findCartList((List<Cart>) redisTemplate.boundHashOps("CART").get(order.getUserId()));        **for** (Cart cart : cartList) {  //订单编号:  **long** id = idWorker.nextId();  ids.add(String.*valueOf*(id));  order.setOrderId(id);  //支付类型: 1: 在线支付(只包含微信支付) 2: 货到付款 页面选择中已经有了  //支付状态  order.setStatus("1");  //不要邮费  order.setPostFee("0");  //来源  order.setSourceType("2");//2:PC  //创建时间  order.setCreateTime(**new** Date());  //更新时间  order.setUpdateTime(**new** Date());  //商家ID  order.setSellerId(cart.getSellerId());  //实付金额  **double** money = 0;  List<OrderItem> orderItemList = cart.getOrderItemList();  **for** (OrderItem orderItem : orderItemList) {  //订单详情表  //详情表主键  orderItem.setId(idWorker.nextId());  //外键  orderItem.setOrderId(id);  money += orderItem.getTotalFee().doubleValue();//金额累加  orderItemDao.insertSelective(orderItem);  }  //实付金额  order.setPayment(**new** BigDecimal(money));    //保存订单  orderDao.insertSelective(order);  }  redisTemplate.boundHashOps("cartList").delete(order.getUserId());  } |

### 2.3.2控制层

修改portal-web的OrderController.java

|  |
| --- |
| /\*\*  \* 提交订单  \* **@author** lx  \*  \*/  @RestController  @RequestMapping("/order")  **public** **class** OrderController {    @Reference  **private** OrderService orderService;  //保存订单  //订单表 订单详情表  @RequestMapping("/add")  **public** Result add(@RequestBody Order order){  **try** {  String userId = SecurityContextHolder.*getContext*().getAuthentication().getName();  order.setUserId(userId);  orderService.add(order);  **return** **new** Result(**true**,"保存订单成功");  } **catch** (Exception e) {  **return** **new** Result(**false**,"保存订单失败");  }  } |

## 2.4前端代码

### 2.4.1服务层

修改portal-web的cartService.js

|  |
| --- |
| //保存订单  **this**.submitOrder=**function**(order){  **return** $http.post('order/add.do',order);  } |

### 2.4.2控制层

修改cartController.js

|  |
| --- |
| //保存订单  $scope.submitOrder=**function**(){  $scope.order.receiverAreaName=$scope.address.address;//地址  $scope.order.receiverMobile=$scope.address.mobile;//手机  $scope.order.receiver=$scope.address.contact;//联系人  cartService.submitOrder( $scope.order ).success(  **function**(response){  **if**(response.success){  //页面跳转  **if**($scope.order.paymentType=='1'){//如果是微信支付，跳转到支付页面  location.href="pay.html";  }**else**{//如果货到付款，跳转到提示页面  location.href="paysuccess.html";  }  }**else**{  alert(response.message); //也可以跳转到提示页面  }  }  );  } |

### 2.4.3页面

修改getOrderInfo.html

|  |
| --- |
| <a class=*"sui-btn btn-danger btn-xlarge"* ng-click=*"submitOrder()"* >提交订单</a> |

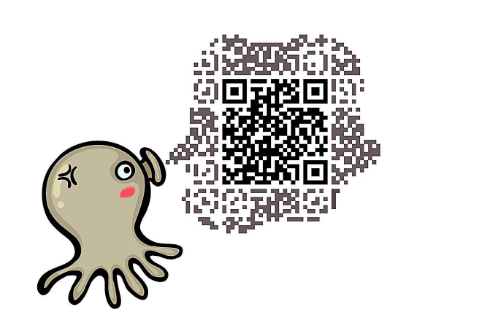
将静态原型中的pay.html paysuccess.html payfail.html 拷贝到portal-web中

# 3.二维码

## 3.1什么是二维码

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。



## 3.2二维码优势

* 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

## 3.3二维码容错级别

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。

## 3.4二维码生成插件qrious

qrious是一款基于HTML5 Canvas的纯JS二维码生成插件。通过qrious.js可以快速生成各种二维码，你可以控制二维码的尺寸颜色，还可以将生成的二维码进行Base64编码。

qrious.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 |

下面的代码即可生成一张二维码

|  |
| --- |
| <html>  <head>  <title>二维码入门小demo</title>  </head>  <body>  <img id="qrious">  <script src="qrious.min.js"></script>  <script>  var qr = new QRious({  element:document.getElementById('qrious'),  size:250, level:'H', value:'http://www.itcast.cn'  });  </script>  </body>  </html> |

运行效果：



大家掏出手机，扫一下看看是否会看到传智播客的官网呢？

# 4.微信扫码支付简介

## 4.1微信扫码支付申请

微信扫码支付是商户系统按微信支付协议生成支付二维码，用户再用微信“扫一扫”完成支付的模式。该模式适用于PC网站支付、实体店单品或订单支付、媒体广告支付等场景。

申请步骤：（了解）

**第一步：注册公众号（类型须为：服务号）**

请根据营业执照类型选择以下主体注册：[个体工商户](http://kf.qq.com/faq/120911VrYVrA151009JB3i2Q.html)| [企业/公司](http://kf.qq.com/faq/120911VrYVrA151013MfYvYV.html)| [政府](http://kf.qq.com/faq/161220eaAJjE161220IJn6zU.html)| [媒体](http://kf.qq.com/faq/161220IFBJFv161220YnqAbQ.html)| [其他类型](http://kf.qq.com/faq/120911VrYVrA151013nYFZ7Z.html)。

**第二步：认证公众号**

公众号认证后才可申请微信支付，认证费：300元/次。

**第三步：提交资料申请微信支付**

登录公众平台，点击左侧菜单【微信支付】，开始填写资料等待审核，审核时间为1-5个工作日内。

**第四步：开户成功，登录商户平台进行验证**

资料审核通过后，请登录联系人邮箱查收商户号和密码，并登录商户平台填写财付通备付金打的小额资金数额，完成账户验证。

**第五步：在线签署协议**

本协议为线上电子协议，签署后方可进行交易及资金结算，签署完立即生效。

本课程已经提供好“传智播客”的微信支付账号，学员无需申请。

## 4.2开发文档

微信支付接口调用的整体思路：

按API要求组装参数，以XML方式发送（POST）给微信支付接口（URL）,微信支付接口也是以XML方式给予响应。程序根据返回的结果（其中包括支付URL）生成二维码或判断订单状态。

在线微信支付开发文档：

<https://pay.weixin.qq.com/wiki/doc/api/index.html>

如果你不能联网，请查阅讲义配套资源 （10配套软件\微信扫码支付）

我们在本章课程中会用到”统一下单”和”查询订单”两组API

1. appid：微信公众账号或开放平台APP的唯一标识
2. mch\_id：商户号 (配置文件中的partner)
3. partnerkey：商户密钥
4. sign:数字签名, 根据微信官方提供的密钥和一套算法生成的一个加密信息, 就是为了保证交易的安全性

## 4.3微信支付SDK

微信支付提供了SDK, 大家下载后打开源码，install到本地仓库。



课程配套的本地仓库已经提供jar包，所以安装SDK步骤省略。

使用微信支付SDK,在maven工程中引入依赖(common中已经引入过了,不用再引入)

|  |
| --- |
| <dependency>  <groupId>com.github.wxpay</groupId>  <artifactId>wxpay-sdk</artifactId>  <version>0.0.3</version>  </dependency> |

我们主要会用到微信支付SDK的以下功能：

1. 获取随机字符串

|  |
| --- |
| WXPayUtil.*generateNonceStr*() |

1. MAP转换为XML字符串（自动添加签名）

|  |
| --- |
| WXPayUtil.*generateSignedXml*(param, partnerkey) |

1. XML字符串转换为MAP

|  |
| --- |
| WXPayUtil.*xmlToMap*(result) |

## 4.4 HttpClient工具类

工具类在“03配置文件\工具类\util\HttpClient.java”,拷贝到common项目中.

HttpClient是Apache Jakarta Common下的子项目，用来提供高效的、最新的、功能丰富的支持HTTP协议的客户端编程工具包，并且它支持HTTP协议最新的版本和建议。HttpClient已经应用在很多的项目中，比如Apache Jakarta上很著名的另外两个开源项目Cactus和HTMLUnit都使用了HttpClient。

HttpClient通俗的讲就是模拟了浏览器的行为，如果我们需要在后端向某一地址提交数据获取结果，就可以使用HttpClient.

关于HttpClient（原生）具体的使用不属于我们本章的学习内容，我们这里这里为了简化HttpClient的使用，提供了工具类HttpClient（对原生HttpClient进行了封装）

HttpClient工具类使用的步骤

|  |
| --- |
| HttpClient client=**new** HttpClient(请求的url地址);  client.setHttps(**true**);//是否是https协议  client.setXmlParam(xmlParam);//发送的xml数据  client.post();//执行post请求  String result = client.getContent(); //获取结果 |

## 4.5准备工作

1. 在common工程中添加工具类HttpClient.java
2. 添加配置文件weixinpay.properties

|  |
| --- |
| appid=wx8397f8696b538317  partner=1473426802  partnerkey=T6m9iK73b0kn9g5v426MKfHQH7X8rKwb  notifyurl=http://a31ef7db.ngrok.io/WeChatPay/WeChatPayNotify |

appid： 微信公众账号或开放平台APP的唯一标识

partner：财付通平台的商户账号

partnerkey：财付通平台的商户密钥

notifyurl: 回调地址

1. 将二维码插件QRious 拷贝到portal-web 的plugins目录中

# 5.品优购-微信支付二维码生成

## 5.1需求分析与实现思路

### 5.1.1需求分析

在支付页面上生成支付二维码，并显示订单号和金额

用户拿出手机,打开微信扫描页面上的二维码,然后在微信中完成支付



### 5.1.2实现思路

我们通过HttpClient工具类实现对远程支付接口的调用。

接口链接：https://api.mch.weixin.qq.com/pay/unifiedorder

具体参数参见“统一下单”API, 构建参数发送给统一下单的url ，返回的信息中有支付url，根据url生成二维码，显示的订单号和金额也在返回的信息中。

## 5.2后端代码实现

### 5.2.1服务接口层

（1）在interface项目下创建下面接口

|  |
| --- |
| **package** com.pinyougou.pay.service;  **import** java.util.Map;  /\*\*  \* 微信支付接口  \* **@author** Administrator  \*  \*/  **public** **interface** WeixinPayService {  /\*\*  \* 生成微信支付二维码  \* **@param** out\_trade\_no 订单号  \* **@param** total\_fee 金额(分)  \* **@return**  \*/  **public** Map createNative(String out\_trade\_no,String total\_fee);  } |

### 5.2.2服务实现层

buyer-service项目下创建下面类

|  |
| --- |
| @Service  **public** **class** WeixinPayServiceImpl **implements** WeixinPayService {  @Value("${appid}")  **private** String appid;    @Value("${partner}")  **private** String partner;    @Value("${partnerkey}")  **private** String partnerkey;    /\*\*  \* 生成二维码  \* **@return**  \*/  **public** Map createNative(String out\_trade\_no,String total\_fee){  //1.创建参数  Map<String,String> param=**new** HashMap();//创建参数  param.put("appid", appid);//公众号  param.put("mch\_id", partner);//商户号  param.put("nonce\_str", WXPayUtil.*generateNonceStr*());//随机字符串  param.put("body", "品优购");//商品描述  param.put("out\_trade\_no", out\_trade\_no);//商户订单号  param.put("total\_fee",total\_fee);//总金额（分）  param.put("spbill\_create\_ip", "127.0.0.1");//IP  param.put("notify\_url", "http://www.itcast.cn");//回调地址(随便写)  param.put("trade\_type", "NATIVE");//交易类型  **try** {  //2.生成要发送的xml  String xmlParam = WXPayUtil.*generateSignedXml*(param, partnerkey);  System.***out***.println(xmlParam);  HttpClient client=**new** HttpClient("https://api.mch.weixin.qq.com/pay/unifiedorder");  client.setHttps(**true**);  client.setXmlParam(xmlParam);  client.post();  //3.获得结果  String result = client.getContent();  System.***out***.println(result);  Map<String, String> resultMap = WXPayUtil.*xmlToMap*(result);  Map<String, String> map=**new** HashMap<>();  map.put("code\_url", resultMap.get("code\_url"));//支付地址  map.put("total\_fee", total\_fee);//总金额  map.put("out\_trade\_no",out\_trade\_no);//订单号  **return** map;  } **catch** (Exception e) {  e.printStackTrace();  **return null**;  }  }  } |

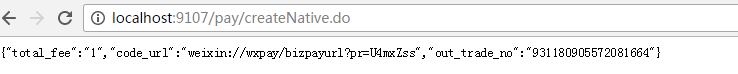
### 5.2.3控制层

portal-web创建PayController.java

|  |
| --- |
| /\*\*  \* 支付控制层  \* **@author** Administrator  \*  \*/  @RestController  @RequestMapping("/pay")  **public** **class** PayController {  @Reference  **private** WeixinPayService weixinPayService;    /\*\*  \* 生成二维码  \* **@return**  \*/  @RequestMapping("/createNative")  **public** Map createNative(){  IdWorker idworker=**new** IdWorker();  **return** weixinPayService.createNative(idworker.nextId()+"","1");  }  } |

这里我们订单号通过分布式ID生成器生成，金额暂时写死，后续开发我们再对接业务系统得到订单号和金额

浏览器测试



## 5.3前端代码实现

### 5.3.1服务层

在portal-web创建 payService.js

|  |
| --- |
| app.service('payService',**function**($http){  //本地支付  **this**.createNative=**function**(){  **return** $http.get('pay/createNative.do');  }  }); |

### 5.3.2控制层

在portal-web 创建payController.js

|  |
| --- |
| app.controller('payController' ,**function**($scope ,payService){  //本地生成二维码  $scope.createNative=**function**(){  payService.createNative().success(  **function**(response){  $scope.money= (response.total\_fee).toFixed(2) ; //金额  $scope.out\_trade\_no= response.out\_trade\_no;//订单号  //二维码  **var** qr = **new** QRious({  element:document.getElementById('qrious'),  size:250,  level:'H',  value:response.code\_url  });  }  );  }  }); |

### 5.3.3页面

修改pay.html ,引入js

|  |
| --- |
| <script type=*"text/javascript"* src=*"plugins/angularjs/angular.min.js"*> </script>  <script type=*"text/javascript"* src=*"js/base.js"*> </script>  <script type=*"text/javascript"* src=*"js/service/payService.js"*> </script>  <script type=*"text/javascript"* src=*"js/controller/payController.js"*> </script>  <script type=*"text/javascript"* src=*"plugins/qrious.min.js"*></script> |

指令

|  |
| --- |
| <body ng-app=*"pinyougou"* ng-controller=*"payController"* ng-init=*"createNative()"*> |

设置二维码图片的ID

|  |
| --- |
| <p class=*"red"*></p>  <div class=*"fl code"*>  <img id=*"qrious"*>  <div class=*"saosao"*>  <p>请使用微信扫一扫</p>  <p>扫描二维码支付</p>  </div>  </div> |

显示订单号

|  |
| --- |
| 订单号：{{out\_trade\_no}} |

显示金额

|  |
| --- |
| <em class=*"orange money"*>￥{{money}}</em>元 |