优供商城交接

2018.3.2

目录

- 业务
  - 基本情况
  - 环境&&账号
    - 测试环境
    - 线上环境
  - 数据来源
    - 商品数据来源
    - 首页数据来源
  - 版本情况
  - 复杂业务逻辑梳理&背景
- 技术
  - 难点逻辑
    - 购物车逻辑
    - model-cart组件（购物车组件）
  - 注意事项
    - pc适配

业务

基本情况

优供商城为链商主线业务，是提供中小超市老板进货的购物平台，目前是基于微信的H5。

环境&&账号

1. 测试环境

地址：http://qa.market-h5.wmdev2.lsh123.com/

账号&密码：
13466640320 000000
发布命令：
fisp release -wmr home -d qa
sh deploy.sh home qa

2. 线上环境

地址：https://m.yougong.elianshang.com/
账号&密码：
19800001117 110011
18612520632 000000

数据来源

商城H5的首页、商品、优惠券等数据皆来自Mis后台配置，以下介绍主要部分：

1. 商品数据

商品数据主要由采购通过 `Mis->商品管理->全部商品` 添加商品，通过 `Mis->商品管理->销售商品` 编辑商品信息；商品创建成功并上架后便可在商城H5中看到。

需要注意数据：

> 起订量 -> moq
> 最大库存 -> inventory_num
> 每日限购 -> day_limit
> 每单限购 -> order_limit

2.首页数据来源

商城首页配置来源主要来自mis运营活动管理，如下表明了首页各个模块对应mis配置的部分。

版本情况

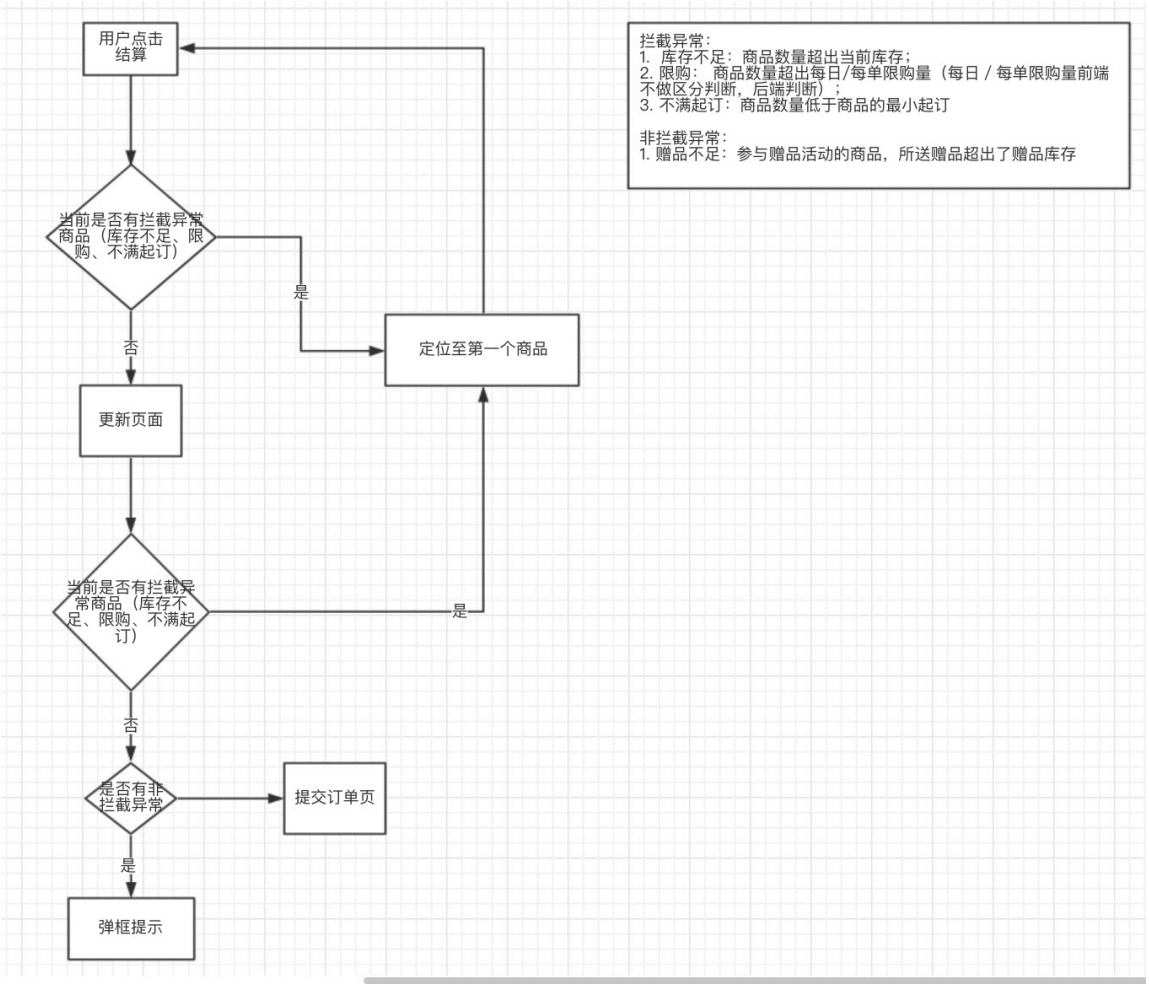
以下为2017年主要大版本以及主要业务展示，具体版本情况见“优供商城版本详情”

> ## v2.4 (2017.12.18)
>
> 账户余额
>
> ## v2.3 （2017.9.4）

> 自由组合套餐
> HTTP
>
> ### v2.2 （2017.7.4）
>
> ### v2.1（2017.6.1）
>
> 语音搜索；
> IM接入（个人中心、订单详情入口）；
>
> ### v2.0（2017.5.5）
>
> 会员身份

复杂业务逻辑梳理&背景

1. 购物车结算时，异常处理逻辑：



技术

难点逻辑

> 注：自定义事件命名规则：以'e'开头，'-'连接各个部分，中间部分为事件定义位置，最后部分语义功能，如'e-main-nav-change'，是在'main-nav'中定义的自定义事件。

- 购物车逻辑
  - 简述
    - 购物车功能
      a. 可编辑，批量删除
      b. 勾选购买
      c. 双购物车（常温 / 冻品分两个tab)
  - 逻辑梳理
    - 购物车大致流程&触发事件梳理，如图“3_购物车逻辑分析”
  - 主要代码分析

```
//widget -> shopping -> cart -> cart.js

var View = Backbone.View.extend({
    //1. 进入购物车页面的入口函数
    initialize: function (options) {
    ...
        //2. 退出编辑态(修改编辑态值,不做其他操作)(场景: 处于编辑态,切换底部tab,回到购物车切换仓库tab,出错)
        //一进入购物车便退出编辑态(1.置标志位 2:清数据)
        CartMainModel.exitEditStatus(true);

        //3. 加载购物车模块的view，渲染购物车页面的主要框架，如头部、底部等
        this.cartView = new CartMainView({thisError : options.thisError });

        //4. 触发切换底部tab事件
        $(document).trigger("e-main-nav-change", "cart");
        //5.触发购物车模块update函数，获取 / 更新购物车数据
        CartMainModel.update({refresh: true, isShowTabs: true});

    ...
    },
    ...
});
```

```
//widget -> cart -> cart.main.model.js
var CartMainModel = Backbone.Model.extend({
    ...
    update: function (opts) {
        opts = opts || {};
        //日志上报，历史原因，用于定位购物车白屏问题，TODO，未解决
        this.setCartRenderLog({f_m_update:new Date().getTime()}
);
        //第一次执行update时，需要利用model-cart组件从localstorage中
获取商品信息。
        // (refresh用于表示第一次执行update，后续在页面内操作仍需要更新
)
        opts.refresh && modelCart.updateLocalSave();

        //获取商品列表
        var carts = modelCart.getAll();

        //历史原因，为兼容旧版本做的遍历
        this.updateBuyCart();

        //设置carts，保存后续使用（注意不做更新渲染）
        this.set("carts", carts);

        //获取当前tab，即常温或是冻品，并设置当前tab，处理同时存在常温，
冻品的购物车的更新（展示当前tab）
        this.getTabs( carts );
        if ( !opts.currentTab ) this.getCurrentTab();

        //获取并设置当前tab下的商品，用于渲染
        this.getItemList( carts );

        //设置cartData，触发View中的渲染
        //设置updatingTotal，部分更新
        this.set({
          cartData: {sku_list: this.get("itemList")},
          updatingTotal: !!opts.updatingTotal
        });


        opts.isShowTabs && this.set({isShowTabs: true});
        // (TODO:不知为何要第二次)
        opts.refresh && this.trigger('e-cart-model-refresh');
    },
  ...
})

//widget -> cart -> cart.main.view.js
```

```
var Cart = Backbone.View.extend({
...
   //非正规render，内包含数据请求
    render: function () {
        ...

        if (currentLength) {
            ...
            that.model.xhr = that.model.fetch({
                type: "POST",
                dataType: "json",
                timeout: 10000,
                data: cartData,
                success: function (_, response) {
                    that.model.setCartRenderLog({f_v_render_aja
x:new Date().getTime()});
                    that._isAbort = false;
                    // 保存请求成功时的本地时间，后面有时间计算，如果一
个用服务器时间，一个用本地时间，会产生问题，毕竟客户机的时间不准确
                    // 所以，两个时间都取客户机上的吧
                    //_.set('lastOpreTime', +new Date());

                    if (response && response.ret === 0 && respo
nse.content && (response.content.block_list.length || response.
content.item_list.length)) {
                        ...
                        // 整合商品数据，把普通商品列表格式化成分组列表
中的最后一个元素
                        //(block_list:分组列表,目前指满减活动列表
                        // item_list: 普通列表)
                        if (content.item_list.length) {
                            content.block_list.push({
                                item_list: content.item_list
                            });
                        }

                        // 保存所有的数据
                        _.set('allItemList', JSON.parse(JSON.st
ringify(content.block_list)));

                        //整合失效商品，值拷贝
                        var list = $.extend( {},that.model.chec
kDisabledItem(content.block_list) );

                        //保存除失效商品以外的商品
                        that.model.set('ableItem', that.model.c
heckDisabledItem(content.block_list).itemList);
```

```
                        var disableItem = list.disabledItemList
;
                        //处理 删除空itemList
                        content.block_list = list.itemList;

                        if(disableItem && disableItem.item_list
){
                            content.block_list.push(disableItem
);
                            that.model.set('disableItem', disab
leItem);
                        }else {
                            that.model.set('disableItem', null)
;
                        }


                        if(!that.model.get('inEditStatus')){
                            //更新优惠券tips
                            that.updateCouponTip( content );
                        }


                        if (that.model.get("updatingTotal")) {
                            // 更新活动标题
                            that.updateGroupTitle(content.block
_list);
                        } else {
                            // 更新整个商品列表
                            that.updateList(content.block_list,
 now);

                        }
                        that.model.setCartRenderLog({f_v_render
_updateList:new Date().getTime()});

                        that.updateGive();
                        //更新头部header,重点更新编辑按钮(当当前购物
车无可编辑商品[失效商品不可编辑],置灰不可点)
                        that.updateHeader();

                        // 是否更新tabs
                        // 只有在切换main-nav的时候才会更新
                        if ( that.model.get("isShowTabs") ) {
                          that.model.set({tabs: {
                            content: content.tabs,
```

```
                            alwaysChange: +new Date()
                        }});
                        that.model.set("isShowTabs", false);
                    }


                    //更新商品的信息（为方便更新小计）
                    $.each(content.block_list, function(k,
v) {
                        modelCart.updateItems(v.item_list);
                    });

                    //  设置reEdStatus,控制恢复编辑态(在更新编辑
态后起效)
                    _.set('reEdStatus', true);

                    //保存money_info
                    _.set('moneyInfo', content.money_info);

                    //  更新总计
                    that.updateTotal(content.money_info, co
ntent.discount_tips);


                    that.model.set('giveData', that.getGive
Data(content.block_list));

                    //只能在此处取完整的异常信息
                    var $errorTips = that.$el.find('.error-
tip');
                    $errorTips = that.model.getSelectedErro
rTip($errorTips);

                    if($errorTips && $errorTips.length){
                            //上报异常数据
                        that.logError($errorTips);

                        //从确认下单页返回定位
                        if(that.thisError){
                            that.goToThisError(that.thisErr
or);

                        //更新数据后定位
                        }else if (that.model.get('goToError
')){
                            that.goToThisError(1);
                        }
```

```
                                //如果点击结算进入的render,并且没有异常时
, 执行下一步
                            }else if(that._fromSubmit){
                                that.realSubmit();
                            }

                        } else if (response && response.ret === 100
021) {
                            Alert.show(response.msg, function() {
                                location.reload();
                            });
                        } else {
                            that.noResponse();
                            Alert.show(response.msg);
                        }
                        ...
                    },
                    ...
                });
            } else {
                ...
            }
           ...
    },
...
});
```

- model-cart组件（购物车组件）
  - 简述
    - 在组件内通过操作_arrBuyCounter，管理加入购物车的商品数据，暴露加减车、获取购物车商品等接口提供给外部业务使用。
  - 代码分析 （注释详尽不做解释）
  - 常件使用
    -

```
    //1.初始化商品数据
        modelCart.init({
            itemList: XXX
        });
        //在生成商品列表时，初始化商品数据
```

```
        //目前主要做的操作有：1.更新商品数据（处理原价与促销价的展示）

    //2.更新商品数据
        modelCart.updateItems({
            itemList: XXX
        })

    //3.获取全部商品列表
        modelCart.getAll()

    //4.加车
        modelCart.add(skuId, {moq:xxx,step_num:xxx}, eventFromT
arget)

    //5.减车
    modelCart.subtract(skuId, {moq:xxx,step_num:xxx}, eventFrom
Target)

    //6.清空购物车
    modelCart.empty()
```

注意事项

- pc适配

  见文档"商城-pc适配方案"

- 商城内混合h5需求开发

  - 背景：由于商城为单页面应用会校验用户登录，当需要开发无需用户登录的H5页面用于嵌入app时需要后端开发单独页面，因此使用再开一个单页面应用的方式，讲所有无需校验用户登录的页面统一管理
  - 位置： widget->cross-h5
  - 目前应用：堆头活动（包括活动列表、活动页、资质审核等）
  - 测试地址：http://qa.market-h5.wmdev2.lsh123.com/home/cross/#tgmarket/list
  - 线上地址：https://m.yougong.elianshang.com/home/cross/#tgmarket/list