# 34 让用户买买买：开发实现商城购物车页面

更新时间：2019-09-06 13:32:29


> 什么是路？就是从没路的地方践踏出来的，从只有荆棘的地方开辟出来的。
>
> —— 鲁迅

上一节我们在商品详情页面实现了单个商品的支付功能，本节将实现在购物车页面同时购买多个商品。

在第二节中，我们已经设计了购物车页面效果，如图 16 所示。

**图 16 购物车页面**


以业务设计、功能设计和页面效果图为基础，按照“分类拆解法”的步骤即可实现购物车页面。

## 1. 拆解步骤

**在本节仅给出拆解结果，拆解过程请回顾第二章第三节对“分类拆解法”的详细讲解内容。**

**请各位同学自己动手实践，按照拆解步骤拆解，结合第一节的业务设计、第二节的功能设计，拆解本节图 16 的购物车页面，然后将自己的拆解结果与本节列出的拆解结果进行对照总结。这是本节内容的实践环节之一。**

购物车页面可以由上到下拆解为 2 个子部件，具体拆解结果如下：

子部件 1 ：购物车清单栏

子部件 1 的显示元素包括：

标题

静态界面，无事件，无数据

清空按钮

单条内容交互界面，事件为用户点击事件，数据为微信客户端的购物车数据缓存

用户点击时间事件的事件响应为：清空购物车数据缓存中的所有商品

购物车商品列表

多条内容交互界面，事件为用户点击事件，数据为微信客户端的购物车数据缓存

在页面加载时，从购物车数据缓存中读取购物车中的商品列表，并显示到页面中

商品列表中的每个商品分左中右三栏水平显示，左侧显示商品缩略图，中间显示商品名称、商品价格，右侧显示删除图标

用户点击商品列表左侧元素、中间元素的事件响应为：页面跳转到商品详情页面，需要向商品详情页面传递商品 ID

用户点击商品列表右侧删除图标的事件响应为：从微信客户端的购物车缓存中去除该商品，并刷新显示购物车商品列表

**商品价格需要根据用户是否付费会员、用户等级是否有购物折扣等条件显示不同内容，具体描述与本章第六节“子部件 2 ：商品价格栏”相同，请参考上一节的内容。**

子部件 2 ：底部操作栏

底部操作栏固定在页面底部显示，不随屏幕滑动变化

当购物车缓存中没有商品数据时，应该隐藏底部操作栏（购物车中没有商品时，无法进行购买）

子部件 2 的显示元素包括：

原价合计

单条内容交互界面，无事件，数据为购物车商品列表中所有商品的原价合计

折扣合计

单条内容交互界面，无事件，数据为原价合计 - 实际支付价格合计

实际支付价格合计

单条内容交互界面，无事件，数据为购物车商品列表中所有商品的原价合计 * 用户享受的折扣率

结算按钮

静态界面，事件为用户点击事件，无数据

用户点击事件的事件响应为：将购物车中所有商品的商品 ID 数组传递给支付云函数，云函数执行完支付逻辑后返回支付结果，在页面中显示云函数返回的支付结果

除上述 2 个子部件外，还需要两个子部件用于显示支付结果：

- 子部件 3 ：显示支付成功结果
- 子部件 4 ：显示支付失败结果

## 2. 编程步骤

与上一节类似，购物车页面需要使用数据服务 `UserService` ， `LevelService` 读取用户信息、计算用户等级，判断用户是否付费会员。

在专栏的源代码中，购物车数据缓存包含了完整的商品信息，商品图片、商品原价等信息可直接从缓存中得到。

**在实际的商城业务中商品信息会发生变化，如商品的价格变动、商品下架等情况时有发生。因此，在使用专栏源代码进行实际商业应用开发时，请仅保存商品 ID 到购物车数据缓存中，在购物车页面中调用数据服务 `ProductService` 来获取商品信息。**

学有余力的同学在实现购物车页面时，可以只保存商品 ID 到购物车数据缓存，商品信息从数据库中读取。

#### 2.1 定义页面子部件及其排列顺序

与上一节类似，我们需要首先理清数据读取的顺序与读取内容。

数据内容包括：

```
data: {
  myInfo: {}, //用户信息，在支付时需要根据当前可用积分判断用户是否有足够积分购买商品
  totalPrice: 0, //购物车中的商品原价合计
  totalDiscountedPrice: 0, //购物车中的商品用户实际支付价格合计
  discount: 1, //用户会员等级或小M卡会员对应的折扣率
  isMembershipExpired: true, //用户是否是小M卡会员（还在会员有效期中）
  showPaySuccess: false, //是否显示支付成功结果
  showPayFaild: false, //是否显示支付失败结果
  cart: [], //购物车数据
},
```

数据读取的顺序为：

在页面加载时，首先读取用户信息计算用户享受的折扣率

```
  /**
   * 生命周期函数--监听页面加载
   */
  onLoad: function(options) {
    //计算用户享受的折扣率
    this.getDiscount()
  },

  /**
   * 根据用户等级与是否付费会员计算用户享受的折扣率
   */
  getDiscount: function() {
    var that = this
    //调用数据服务获取用户的用户等级与付费会员有效期
    levelService.getLevelList(
      //获取成长体系中的所有成长等级回调函数
      function(levelList) {
        var levels = levelList
        userService.getUserInfo(
          //获取用户信息回调函数
          function(userinfo) {
            var myInfo = userinfo
            //获取用户等级
            var myLevel = levels.filter(e => e.minGrowthValue <= myInfo.growthValue && myInfo.growthValue <= e.max
GrowthValue)[0]
            //根据付费会员有效期计算用户是否是小M卡会员
            var isMembershipExpired = myInfo.memberExpDate < new Date()
            //设置用户是否是小M卡会员的标志
            that.setData({
              isMembershipExpired: isMembershipExpired,
              myInfo: myInfo
            })
            if (!isMembershipExpired) {
              //如果用户是在有效期的小M卡会员，设置小M卡会员折扣
              that.setData({
                discount: MEMBERSHIPDISCOUNT
              })
            } else if (myLevel.bonus.length == 3) {
              //如果用户不是小M卡会员，设置用户当前等级对应的折扣
              that.setData({
                discount: myLevel.bonus[1].discount
              })
            }
            //读取购物车中商品数据
            that.initCart()
            //计算购物车每个商品的价格
            that.calPrice()
          })
      })
  },
```

从微信客户端的购物车数据缓存中读取购物车商品数据

购物车数据存储在微信客户端的数据缓存中，使用小程序官方提供的数据缓存 API 读取（数据缓存 API 的使用方法请回顾本章第四节内容）：

```
  /**
   * 从微信客户端的购物车数据缓存中读取购物车商品数据
   */
  initCart: function() {
    var value = wx.getStorageSync('cart');
    if (value) {
      this.setData({
        cart: value
      });
    }
  },
```

根据用户享受的折扣率计算购物车中每个商品的价格

```
/**
 * 根据用户享受的折扣率计算购物车中每个商品的价格
 */
calPrice() {
  //购物车中所有商品原价合计
  var totalPrice = 0
  //购物车中所有商品折扣价合计
  var totalDiscountedPrice = 0
  var cart = this.data.cart
  for (var i in this.data.cart) {
    cart[i].price = parseInt(cart[i].price) //商品原价
    cart[i].discountedPrice = Math.ceil(cart[i].price * this.data.discount) //根据折扣计算商品折扣价
    totalPrice += cart[i].price //原价合计
    totalDiscountedPrice += Math.ceil(cart[i].price * this.data.discount) //折扣价合计
  }
  //设置商品价格数据
  this.setData({
    cart: cart,
    totalDiscountedPrice: totalDiscountedPrice,
    totalPrice: totalPrice
  });
},
```

**请参照前面章节的内容自行添加数据服务引用。**

在 WXML 页面模板中我们需要定义在什么条件下显示哪些子部件，除与上一节类似的支付结果显示隐藏外，底部操作栏在购物车缓存中没有商品数据时应该隐藏。

```
<!-- 不显示支付结果时，才显示子部件 1、2 -->
<view wx:if="{{!showPaySuccess && !showPayFaild}}">
  <!-- 子部件 1 ：购物车清单栏 -->
  <view class="weui-panel weui-panel_access">
  </view>
  <!-- 子部件 2 ：底部操作栏 购物车中有商品时才显示-->
  <view wx:if="{{cart.length > 0}}" class="cu-bar bg-white tabbar border shop bottom_pay_button">
  </view>
</view>

<!-- 当支付接口返回支付成功时，显示支付成功结果 -->
<!-- 子部件 3 ：显示支付成功结果 -->
<view wx:if="{{showPaySuccess}}" class="weui-msg bg-white padding">
</view>

<!-- 当支付接口返回支付失败时，显示支付失败结果 -->
<!-- 子部件 4 ：显示支付失败结果 -->
<view wx:if="{{showPayFaild}}" class="weui-msg bg-white padding">
</view>
```

**2.2 实现子部件 1 ：购物车清单栏**

子部件 1 的标题与清空按钮在一行左右显示，使用 WeUI 的 Flex 组件实现布局。清空按钮使用 ColorUI 的图标元素 `icon-delete` ：

```
<view class="weui-panel__hd">
    <view class="weui-flex">
        <!-- 标题 -->
        <view class="weui-flex__item">
            购物车
        </view>
        <!-- 清空按钮 -->
        <view class="weui-flex__item text-right">
            <text class="icon-delete" bindtap='clearCart'></text>
        </view>
    </view>
</view>
```

清空按钮的点击事件为清空购物车数据缓存中的所有商品：

```
/**
 * 清空购物车数据
 */
clearCart: function() {
  //清空购物车的显示数据
  this.setData({
    cart: []
  });
  //缓存中清除购物车数据
  wx.removeStorage({
    key: 'cart',
  })
},
```

购物车商品列表使用条件渲染显示。

列表中的每个商品分左中右三栏水平显示，使用 WeUI 的 Flex 组件，设置中间元素为 `weui-flex__item` 即可实现布局。

```
<!-- 商品列表 -->
<view class="weui-panel__bd">
 <!-- 使用条件渲染显示购物车中的每个商品 -->
 <block wx:for="{{cart}}" wx:key="{{item.index}}">
<view class="weui-cell weui-cell_access">
 <view class="weui-cell__bd">
  <!-- 每个商品分左中右三栏水平显示，使用 WeUI 的 Flex 组件实现 -->
 <view class="weui-flex">
  <!-- 左侧栏显示商品图片 -->
  <!-- 左侧栏点击后页面跳转商品详情页面 -->
  <navigator url="../product/product?index={{item.index}}">
 <image class="product_thumbnail_image" src="{{item.smallcoverimg}}" />
  </navigator>
  <!-- 中间栏显示商品信息 -->
      <!-- 在中间栏设置class="weui-flex__item"实现左中右三栏水平显示效果 -->
  <!-- 中间栏点击后页面跳转商品详情页面 -->
  <navigator url="../product/product?index={{item.index}}" class="weui-flex__item margin-lr-sm">
  <view class="text-lg">{{item.bookname}}</view>
  <view class="text-sm">
   <!-- 与本章第六节类似，根据不同的条件显示不同的商品价格 -->
   <block wx:if="{{isMembershipExpired && discount < 1}}">
  <view class="text-red">
   折扣价 p{{item.discountedPrice}}
  </view>
  <view>
   <text class="line_through"> 原价 p{{item.price}} </text>
  </view>
   </block>
   <block wx:if="{{!isMembershipExpired}}">
  <view class="text-red">
   小M卡专享价 p{{ item.discountedPrice }}
  </view>
  <view>
   <text class="line_through"> 原价 p{{item.price}} </text>
  </view>
   </block>
   <block wx:if="{{isMembershipExpired && discount == 1}}">
  <view class="text-red">
   限时特价 p{{item.discountedPrice}}
  </view>
   </block>
  </view>
  </navigator>
  <!-- 右侧栏显示删除图标-->
  <view class="text-xs">
  <text class="icon-delete" bindtap='deleteProductFromCart' data-item="{{item}}"></text>
  </view>
 </view>
  </view>
</view>
 </block>
</view>
```

右侧删除图标的点击事件为：从微信客户端的购物车缓存中去除该商品，并刷新显示购物车商品列表：

```
/**
 * 删除购物车数据
 */
deleteProductFromCart: function(event) {
  //获取用户点击的是哪个商品的删除图标
  var productIndex = event.currentTarget.dataset.item.index
  //从缓存读取购物车数据
  var value = wx.getStorageSync('cart');
  if (value) {
    //在缓存数据中查找到要删除的商品
    var product = value.filter(e => e.index == productIndex)
    if (product.length > 0) {
      //得到要删除的商品在购物车缓存中的索引号
      var index = value.indexOf(product[0])
      if (index > -1) {
        //根据索引号在购物车缓存中删除商品
        value.splice(index, 1)
        //更新购物车的显示数据
        this.setData({
          cart: value
        });
        //将最新的购物车数据更新到本地缓存
        wx.setStorage({
          key: "cart",
          data: value,
        })
        //在删除购物车中的商品后，需要重新计算商品原价合计、折扣价合计
        this.refreshPrice()
      }
    }
  }
},
```

**2.3 实现子部件 2 ：底部操作栏**

与第六节的底部操作栏类似，样式可以在 ColorUI 的操作条组件中找到，简单修改 ColorUI Demo 小程序的代码即可实现 WXML 页面模板：

```
<!-- 子部件 2 ：底部操作栏 购物车中有商品时才显示-->
<view wx:if="{{cart.length > 0}}" class="cu-bar bg-white tabbar border shop bottom_pay_button">
  <view class="action">
    <view>总计</view>
    <text class="text-red">p{{totalPrice}}</text>
  </view>
  <view class="action">
    <view>折扣</view>
    <text class="text-red">p{{totalPrice - totalDiscountedPrice}}</text>
  </view>
  <view class="action">
    <view>实际支付</view>
    <text class="text-red">p{{totalDiscountedPrice}}</text>
  </view>
  <view bindtap='onPayButtonClick' class="bg-red submit">结算</view>
</view>
```

使用自定义样式 `bottom_pay_button` 实现固定在页面底部显示。

子部件 2 中结算按钮点击事件的实现逻辑与实现代码，与第七章第五节付费会员支付按钮完全一致，区别仅在于调用的支付接口名称不同，传递的参数不同。

在结算按钮点击事件中传递的参数是商品 ID 数组，可从购物车的商品数据中得到：

```
//商品 ID 数组
var productsIndex = []
for (var i in that.data.cart) {
  //将购物车每个商品的商品 ID 添加到商品 ID 数组
  productsIndex.push(that.data.cart[i].index)
}
```

请各位同学参考第七章第五节的代码自行实现结算按钮的点击事件 `onPayButtonClick` ，也可参考专栏源代码的 `cart.js` 文件，具体代码位置见本节末尾图 17。

**2.4 实现子部件 3 与子部件 4**

子部件 3 与 子部件 4 的实现请参考第七章第五节 “2.7 实现子部件 6 与 子部件 7 ：显示支付结果” 或专栏源代码。

**由于篇幅所限，包含完整样式的 WXML 页面模板代码请查阅专栏源代码 cart.wxml 与 cart.wxss ，完整的 JS 逻辑代码见 cart.js ，具体代码位置见本节末尾图 17 。**

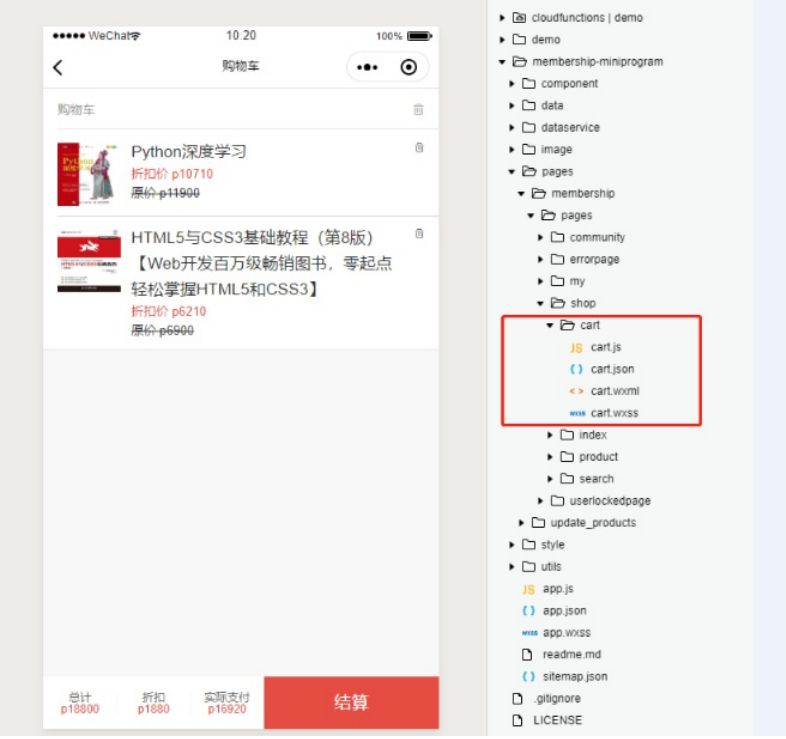
## 3. 专栏源代码

本专栏源代码已上传到 GitHub ，请访问以下地址获取：

https://github.com/liujiec/Membership-ECommerce-Miniprogram

本节源代码内容在图 17 红框标出的位置。

**图 17 本节源代码位置**




## 下节预告

下一节，我们将实现商城的最后一个功能页面，订单列表页面。

## 实践环节

实践是通往大神之路的唯一捷径。

本节实操内容：

- 请结合第二章第三节对“分类拆解法”的详细讲解内容，拆解本节图 **16** 的购物车页面，然后将自己的拆解结果与本节列出的拆解结果进行对照总结。
- 编写代码完成购物车页面，如碰到问题，请阅读本专栏源代码学习如何实现。

}

33 让用户掏钱：开发实现商城商品详情页面

35 给用户看账单：开发实现订单列表页面