32 准备好收银机：开发实现商城支付接口

更新时间：2019-09-02 10:47:50

> 成功＝艰苦的劳动＋正确的方法＋少谈空话。
>
> ——爱因斯坦

前两节我们完成了商城首页和商品搜索页面的开发，在继续后续页面开发之前，我们需要先编写云函数实现商品购买的支付接口。

在本章第二节中，我们已经整理了支付接口的功能，包括：

- 功能点 1：同时支持单个商品的支付（商品详情页面）与多个商品的支付（购物车页面）
- 功能点 2：为避免黑客伪造数据进行攻击，在支付接口中不能直接使用客户端计算的价格。支付接口需要根据用户购买的商品、用户等级、是否付费会员等信息，重新计算用户实际需要支付的积分；
- 功能点 3：校验用户的当前可用积分是否足够支付，如积分不足，接口返回支付失败，失败原因为积分不足；
- 功能点 4：在订单记录表中记录本次支付信息，供用户在订单列表中查看；
- 功能点 5：在商品购买记录表中记录用户该笔订单购买的每一个商品的信息；
- 功能点 6：在积分变动记录表中增加本次支付对应的积分变动记录；
- 功能点 7：在成长值获取记录表中增加因购买商品获得的成长值记录；
- 功能点 8：修改用户的当前可用积分为原当前可用积分 - 本次支付消耗积分，修改用户的当前总成长值为原当前总成长值 + 本次支付消耗积分。

根据以上功能设计，结合第七章第四节实现付费会员支付接口的经验，我们可以整理出商城支付接口的支付程序逻辑。

1. 商城支付接口程序逻辑

根据功能点 1 和 功能点 2 我们可以整理出支付接口需要的输入参数：

- 购买商品的用户的 OpenID，该参数可以直接使用云函数的 `cloud.getWXContext()` 得到

- 用户购买商品 ID 数组 `productsIndex` （在购物车中可同时购买多个商品），该参数由小程序客户端调用云函数时提供

根据功能点 3 我们可以整理出支付接口的返回结果：

- 支付结果：bool 值表示成功（true）或失败（false）
- 失败原因：当支付结果是 false 时，需要返回失败原因

具体的程序逻辑如下：

- 功能点 2 ：
  - 从用户表 `user` 中获取用户当前总成长值 `growthValue` 与 付费会员到期时间 `memberExpDate`
  - 从用户等级与等级特权表 `level` 表中获取所有的用户等级信息 `levels`
  - 根据用户当前总成长值 `growthValue` ，从用户等级信息 `levels` 中查询出用户的当前等级 `userLevel`
  - 如果用户的付费会员还未到期（ `memberExpDate` 大于等于当前时间），设置购买商品的折扣率 `discount` 为付费会员享受的 7 折折扣率
  - 如果用户的付费会员已过期（ `memberExpDate` 大于当前时间），从用户的当前等级 `userLevel` 中得到购买商品的折扣率 `discount`
  - 从商品信息表 `product` 中获取用户购买商品 ID 数组 `productsIndex` 中所有商品的信息 `products`
  - 计算 `products` 中所有商品的原价合计 `totalPrice`
  - 计算用户购买所有商品实际需要支付的积分合计 `paymentFee` （由于积分的获取是整数，积分的支付也应该设置为整数，计算每个商品的实际支付价格时需要进行取整处理：实际支付积分 = 向上取整 (商品原价 * discount) ）
- 功能点 3 ：
  - 从用户表 `user` 中获取用户当前可用积分 `point` ，并校验用户的当前可用积分是否足够购买所有商品，如果积分不足则返回支付失败并设置失败原因为 "很抱歉，你的积分不足，无法购买"
- 功能点 4 ：
  - 向订单记录表 `user_order` 中插入一条新记录，记录本次支付的商品与订单信息（ `discount` 、 `totalPrice` 、 `paymentFee` 等）
  - 插入记录后可得到插入的订单记录 ID `orderId`
- 功能点 5 ：
  - `products` 中的每一个商品都需要在商品购买记录表中插入一条新记录 `user_paid` ， `orderId` 在功能点 4 中得到，折扣率 `discount` 在功能点 2 中已计算出，商品实际支付价格需要按功能点 2 描述的取整方式进行处理
- 功能点 6 ：
  - 向积分变动记录表 `user_point` 中插入一条新记录，记录本次支付的积分变动信息
- 功能点 7 ：
  - 用户购买商品获得的成长值即用户购买商品支付的积分数 `paymentFee`
  - 向成长值获取记录表 `user_growth_value` 中插入一条新记录，记录本次支付获得的成长值信息
- 功能点 8：

- 计算用户购买商品后新的用户当前总成长值 `newGrowthValue` = 用户原当前总成长值 `growthValue` + 用户购买商品支付的积分 `paymentFee`
- 在用户表 `user` 中更新用户的当前总成长值为 `newGrowthValue`
- 计算用户购买商品后新的当前可用积分 `newPoint` = 用户原当前可用积分 `point` - 用户购买商品支付的积分 `paymentFee`
- 在用户表 `user` 中更新用户的当前可用积分为 `newPoint`

2 商城支付接口代码实现

如还未在云数据库中创建订单记录表 `user_order` 与商品购买记录表 `user_paid` 的同学，请先在云数据中新建该表。

根据已经整理出的输入输出参数与程序逻辑，我们就可以新建商城支付接口的云函数 `payProduct` ，然后在 index.js 中实现支付逻辑：

```
const cloud = require('wx-server-sdk')

// 初始化 cloud
cloud.init()
const db = cloud.database()
const _ = db.command

//小M卡会员享受的付费会员折扣率
const MEMBERSHIPDISCOUNT = 0.7

// 云函数入口函数
/**
 * 商城支付接口，支付积分购买商品
 * @param {data} 要购买的商品信息
 * {
 *    data:{
 *      productsIndex,    //[] 用户购买的商品的商品ID数组
 *    }
 * }
 * @return {object} 支付结果
 * {
 *    data,    //bool 支付成功或失败
 *    errMsg   //如果支付失败，该字段包含支付失败的具体错误信息
 * }
 */
exports.main = async(event, context) => {
  const wxContext = cloud.getWXContext()
  //设置支付接口返回结果的默认值
  var result = false
  var errMsg = ''

//----------------------------------------功能点 2 ----------------begin
  //折扣率
  var discount = 1
  //读取用户信息
  var user = (await db.collection('user')
    .where({
      //云函数是在服务端操作，对所有用户的数据都有操作权限
      //在云函数中查询用户数据，需要添加openid的查询条件
      _openid: wxContext.OPENID
    })
    .get()).data[0]
  //读取用户等级信息
  var levels = (await db.collection('level').get()).data
  //查询出用户的当前等级
  var userLevel = levels.filter(e => e.minGrowthValue <= user.growthValue && user.growthValue <= e.maxGrowthValue)[0]
  //判断用户是否是付费会员
  var isMembershipExpired = user.memberExpDate < new Date()
  if (!isMembershipExpired) {
    //如果用户的付费会员未过期，设置小M卡会员折扣
    discount = MEMBERSHIPDISCOUNT
```

```
  } else if (userLevel.bonus.length == 3) {
    //如果用户的付费会员已过期，设置用户等级对应的折扣
    discount = userLevel.bonus[1].discount
  }
  //根据商品ID数组，读取商品信息
  var products = (await db.collection('product')
    .where({
      index: _.in(event.productsIndex)
    })
    .get()).data
  //原价合计
  var totalPrice = 0
  //实际支付价格合计
  var paymentFee = 0
  //计算合计价格
  for (var i in products) {
    totalPrice += parseInt(products[i].price)
    //实际支付价格 = 向上取整(商品原价 * discount)
    paymentFee += Math.ceil(parseInt(products[i].price) * discount)
  }
//----------------------------------------功能点 2 ----------------end

  if (user.point < paymentFee) {
    //功能点 3 ：如果积分不足则返回支付失败并设置失败原因
    errMsg = "很抱歉，你的积分不足，无法购买"
  } else {

//----------------------------------------功能点 4 ----------------begin
    //向订单记录表中插入一条新记录，记录本次支付的商品与订单信息
    //插入记录后获得插入的订单记录 ID
    var orderId = (await db.collection('user_order')
      .add({
        data: {
          _openid: wxContext.OPENID, //云函数添加数据不会自动插入openid，需要手动定义
          date: db.serverDate(), //购买商品时间
          productsIndex: event.productsIndex, //用户购买的商品 ID 数组
          totalPrice: totalPrice, //商品原价合计
          paymentFee: paymentFee, //折扣率
          discount: discount, //实际支付价格合计
          status: 'paid'
        }
      })).id
//----------------------------------------功能点 4 ----------------end

//----------------------------------------功能点 5 ----------------begin
    //在商品购买记录表中插入用户购买的每一个商品的记录
    for (var i in products) {
      await db.collection('user_paid')
        .add({
          data: {
            _openid: wxContext.OPENID, //云函数添加数据不会自动插入openid，需要手动定义
            date: db.serverDate(), //购买商品时间
            productIndex: products[i].index, //购买的商品 ID
            price: parseInt(products[i].price), //商品原价
            paymentFee: Math.ceil(parseInt(products[i].price) * discount), //实际支付价格 = 向上取整(商品原价 * discount)
            discount: discount, //折扣率
            orderId: orderId //对应的订单记录 ID
          }
        })
    }
//----------------------------------------功能点 5 ----------------end

//----------------------------------------功能点 6 ----------------begin
    //向积分变动记录表插入一条新记录，记录本次支付的积分变动信息
    await db.collection('user_point')
      .add({
        data: {
          _openid: wxContext.OPENID, //云函数添加数据不会自动插入openid，需要手动定义
          date: db.serverDate(), //积分变动时间
          changePoints: -1 * paymentFee, //积分变动值，消耗积分为负值
          operation: "购买商品", //积分变动原因
          timestamp: '',
          orderId: orderId //对应的订单记录 ID
```

```
      }
    })
//----------------------------------------功能点 6 ----------------end

//----------------------------------------功能点 7 ----------------begin
  //向成长值获取记录表中插入一条新记录，记录本次支付获得的成长值信息
  await db.collection('user_growth_value')
    .add({
      data: {
        _openid: wxContext.OPENID, //云函数添加数据不会自动插入openid，需要手动定义
        date: db.serverDate(), //获取成长值时间
        changeGrowthValue: paymentFee, //获得的成长值，即用户购买商品支付的积分数
        operation: "购买商品", //成长值来源
        timestamp: '',
        orderId: orderId, //对应的订单记录 ID
        noteId: ''
      }
    })
//----------------------------------------功能点 7 ----------------end

//----------------------------------------功能点 8 ----------------begin
  //用户购买商品后新的当前可用积分
  var newPoint = user.point - paymentFee
  //用户购买商品后新的用户当前总成长值
  var newGrowthValue = user.growthValue + paymentFee
  //在用户表中更新用户的当前可用积分、当前总成长值
  var updateUserResult = await db.collection('user')
    .where({
      //云函数是在服务端操作，对所有用户的数据都有操作权限
      //在云函数中查询用户数据，需要添加openid的查询条件
      _openid: wxContext.OPENID
    })
    .update({
      data: {
        point: newPoint, //新的用户当前可用积分
        growthValue: newGrowthValue //新的用户当前总成长值
      }
    })
  if (updateUserResult.stats.updated == 1) {
    result = true
  } else {
    errMsg = "支付异常，如有疑问请联系客服"
  }
//----------------------------------------功能点 8 ----------------end
 }

 //返回支付结果
 return {
  data: result,
  errMsg: errMsg
 }
}
```

在编写完云函数后，我们可以使用云开发提供的云函数测试功能，测试云函数编写是否正确。

微信官方的“小程序开发文档”中有云函数测试功能的详细说明，具体位置为：

云开发 -> 开发指引 -> 云函数 -> 测试、日志与监控。

请阅读说明文档，然后测试你编写的云函数。

> Tips：
>
> 在第七章第四节和本节中，通过两个支付接口的实现过程，向各位同学介绍了新人编写后端代码最稳妥的方法：

1. 整理出后端业务功能点
2. 针对每个业务功能点，使用自然语言（中文、英文、日文等）把程序逻辑写出来
3. 将自然语言描述的程序逻辑 “翻译” 为编程语言

这个方法步骤并不是我创造的，日本的软件项目开发就是这么干的，请放心使用：）

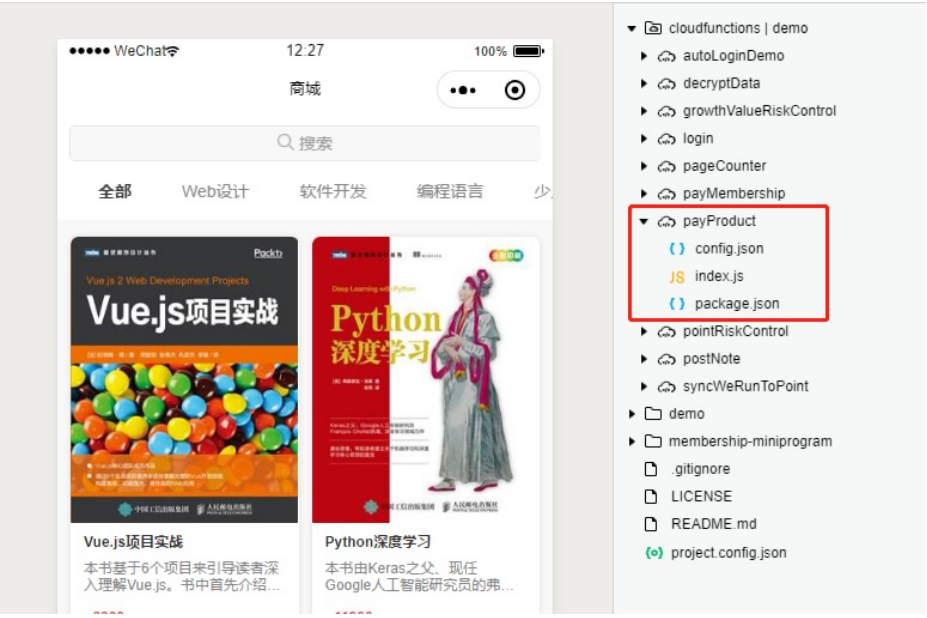
3. 专栏源代码

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 13 红框标出的位置。

图 13 本节源代码位置



下节预告

下一节，我们将实现商品详情页面，并调用本节编写的云函数实现商品购买功能。

实践环节

实践是通往大神之路的唯一捷径。

本节实操内容：

- 编写代码完成商城支付接口的完整云函数，如碰到问题，请阅读本专栏源代码学习如何实现。
- 阅读云开发 -> 开发指引 -> 云函数 -> 测试、日志与监控，然后测试云函数是否能返回正常结果。

}

31 搜索热词、搜索历史和联想搜索

33 让用户掏钱：开发实现商城商品详情页面