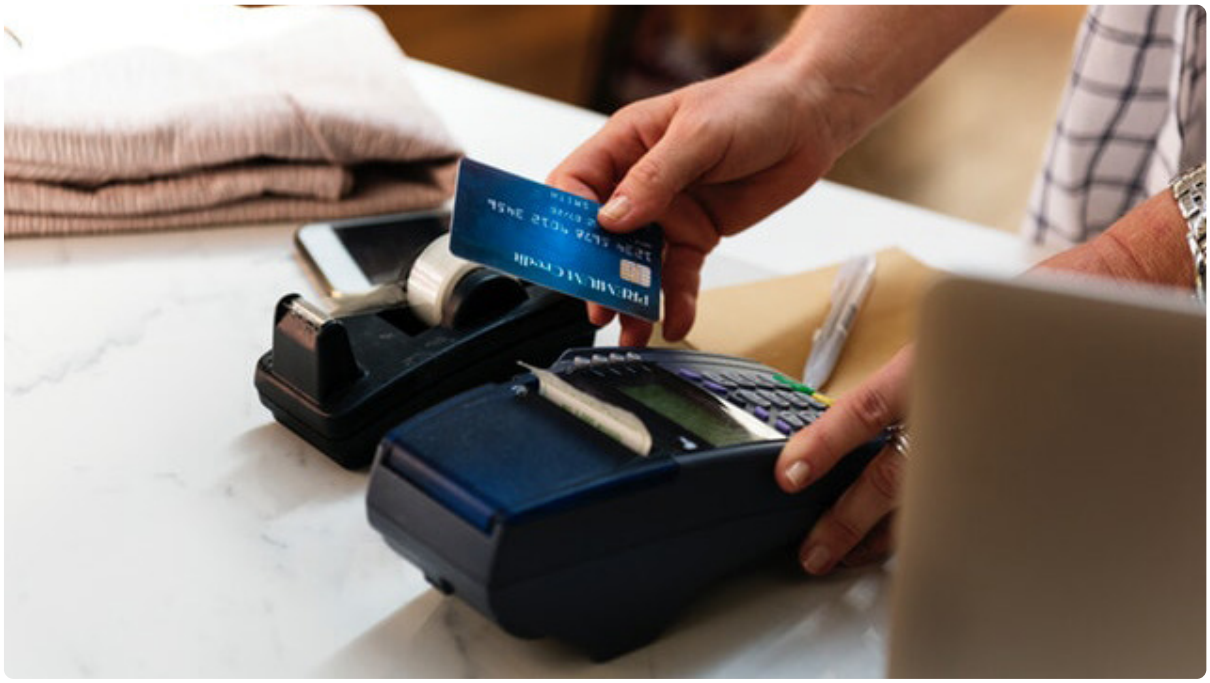
26 准备好收银机：开发实现付费会员支付接口

更新时间：2019-08-19 14:26:41



> 耐心和恒心总会得到报酬的。
>
> ——爱因斯坦

上一节我们已经完成了付费会员首页面的开发，在本节和下一节中将讲解付费会员套餐购买页面的详细实现过程。

本节将使用云函数实现付费会员支付接口，下一节实现付费会员套餐购买页面内容。

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 ：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付费会员套餐和用户等级，重新计算用户实际需要支付的积分；
- 功能点 2 ：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 3 ：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 4 ：在付费会员套餐购买记录表中记录本次支付信息，供对账等后续业务使用；
- 功能点 5：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；
- 功能点 6 ：根据用户购买的付费会员套餐，增加用户的付费会员有效期。

首先，我们要根据功能清单设计出付费会员支付接口的程序逻辑。

1. 付费会员支付接口程序逻辑

根据功能点 1 我们可以整理出支付接口需要的输入参数：

- 购买套餐的用户的 OpenID，该参数可以直接使用云函数的 `cloud.getWXContext()` 得到；
- 用户购买的套餐的 ID，该参数由小程序客户端调用云函数时提供。

根据功能点 2 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 1：

- 从用户表 `user` 中获取用户当前总成长值 `growthValue` ；
- 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`；
- 从用户的当前等级 `userLevel` 中得到购买付费会员的折扣积分 `discount`；
- 从付费会员套餐表 `membership_plan` 中根据套餐 ID 得到用户购买的套餐信息 `membershipPlan`；
- 计算用户购买套餐实际需要支付的积分 `paymentFee = membershipPlan.price - discount`。

功能点 2：

- 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套餐，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 3：

- 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 4：

计算用户购买套餐后新的会员有效期：

- 如果用户是非会员（即用户表 `user` 中的 `memberExpDate` 小于等于当前时间），则设置付费会员生效时间 `fromDate` 为当前时间（即用户购买的套餐从现在开始生效）；
- 如果用户是付费会员（即用户表 `user` 中的 `memberExpDate` 大于当前时间），则设置付费会员生效时间 `fromDate` 为 `memberExpDate` （即用户购买的套餐在以前购买的套餐失效后才开始生效）；
- 用户新的付费会员有效期 `newMemberExpDate` = 付费会员生效时间 `fromDate` + 购买套餐获得的会员有效期 `validity` * 30 天。

向付费会员套餐购买记录表 `user_membership` 中插入一条新记录，记录本次支付购买的套餐信息。

功能点 5 与功能点 6：

- 计算用户购买套餐后新的当前可用积分 `newPoint` = 用户原当前可用积分 `point` - 用户购买套餐支付的积分 `paymentFee`；
- 在用户表 `user` 中更新用户的当前可用积分为 `newPoint` ，付费会员有效期为 `newMemberExpDate` 。

2 付费会员支付接口代码实现

如还未在云数据库中创建付费会员套餐购买记录表 `user_membership` 的同学，请先在云数据中新建该表。

在小程序客户端中我们统一了云数据库的访问方式：通过数据服务的方式来访问云数据库。

在云函数中我们可以直接访问数据库，可同样可以在云函数中创建数据服务，通过数据服务来访问数据库。

由于专栏项目在云函数中实现的业务逻辑不多，专栏的云函数源代码是采用直接访问数据库的方式。**如果在实际的商业项目中要在云函数中实现很多业务逻辑或有很多的数据库访问代码，建议大家也在云函数中创建数据服务，通过数据服务来访问数据库。**

将数据库访问方法封装在单独的数据服务层中可以让我们的代码更便于维护（分层的代码结构，可读性更强、需要编写的代码量也更少）、复用性也更强（增加新功能时只需要一行代码调用原有数据服务接口，而不需要重新实现数据库访问代码），**特别是对需要长期迭代的项目和大型项目而言，分层、复用尤其重要。**

根据我们已经整理出的输入输出参数与程序逻辑，我们就可以新建付费会员支付接口的云函数 `payMembership` ，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

// 云函数入口函数
/**
 * 支付积分购买会员套餐
 * @param {data} 要购买的会员套餐
 * {
 *    data:{
 *      membershipPlanId, //用户购买的套餐
 *    }
 * }
 * @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 1 ----------------begin
  //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]
  //得到购买付费会员的折扣积分
  var discount = userLevel.bonus.length == 3 ? userLevel.bonus[2].discount : 0
  //读取用户购买的会员套餐
  var membershipPlan = (await db.collection('membership_plan')
    .where({
      id: event.membershipPlanId
    })
    .get()).data[0]
  //计算用户购买套餐实际需要支付的积分
  var paymentFee = membershipPlan.price - discount
//----------------------------------------功能点 1 ----------------end

  if (user.point < paymentFee) {
    //功能点 2 ：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 3 ----------------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: "购买会员", //积分变动原因
        timestamp: '',
        orderId: ''
      }
    })
//----------------------------------------功能点 3 ----------------end

//----------------------------------------功能点 4 ----------------begin
  var memberExpDate = user.memberExpDate //用户原付费会员有效期
  var validity = membershipPlan.validity //购买套餐获得的会员有效期月数（1月 = 30天）
  //如果用户是非会员，用户购买的套餐从现在开始生效
  var fromDate = new Date()
  if (memberExpDate > fromDate) {
    //如果用户是付费会员，用户购买的套餐在以前购买的套餐失效后才开始生效
    fromDate = memberExpDate
  }
  //用户新的付费会员有效期
  var newMemberExpDate = new Date(fromDate.getTime() + 1000 * 60 * 60 * 24 * 30 * validity)
  //向付费会员套餐购买记录表插入一条新记录，记录本次支付购买的套餐信息
  await db.collection('user_membership')
    .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(), //购买付费会员套餐时间
        membershipPlanId: event.membershipPlanId, //付费会员套餐ID
        userLevel: userLevel.id, //用户当前等级ID
        price: membershipPlan.price, //套餐原价
        discount: discount, //购买付费会员的折扣积分
        paymentFee: paymentFee, //实际支付价格
        fromDate: fromDate, //付费会员生效时间
        toDate: newMemberExpDate //付费会员失效时间
      }
    })
//----------------------------------------功能点 4 ----------------end

//----------------------------------------功能点 5 ----------------begin
  //计算用户购买套餐后新的当前可用积分
  var newPoint = user.point - paymentFee
  //更新用户的当前可用积分 和 付费会员有效期
  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: {
        memberExpDate: newMemberExpDate, //支付后用户的付费会员有效期
        point: newPoint //支付后用户的当前可用积分
      }
    })
  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 5 ----------------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。

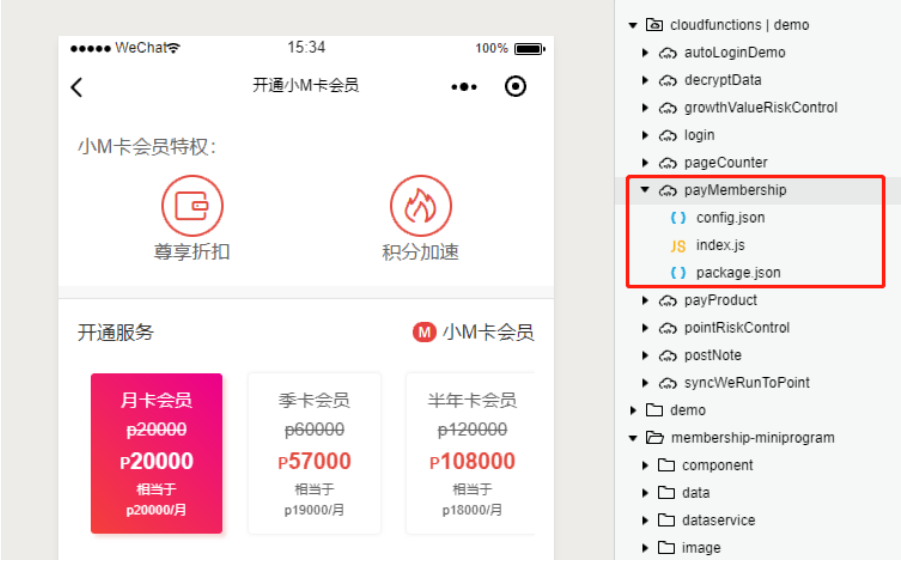
3. 专栏源代码

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 7 红框标出的位置。

图 7 本节源代码位置



下节预告

下一节，我们将调用本节编写的云函数，实现付费会员套餐购买页面。

实践环节

实践是通往大神之路的唯一捷径。

本节实操内容：

- 编写代码完成付费会员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。

}

25 发个卡给用户：开发实现付费会员首页面

27 让用户掏钱：开发实现付费会员支付页面