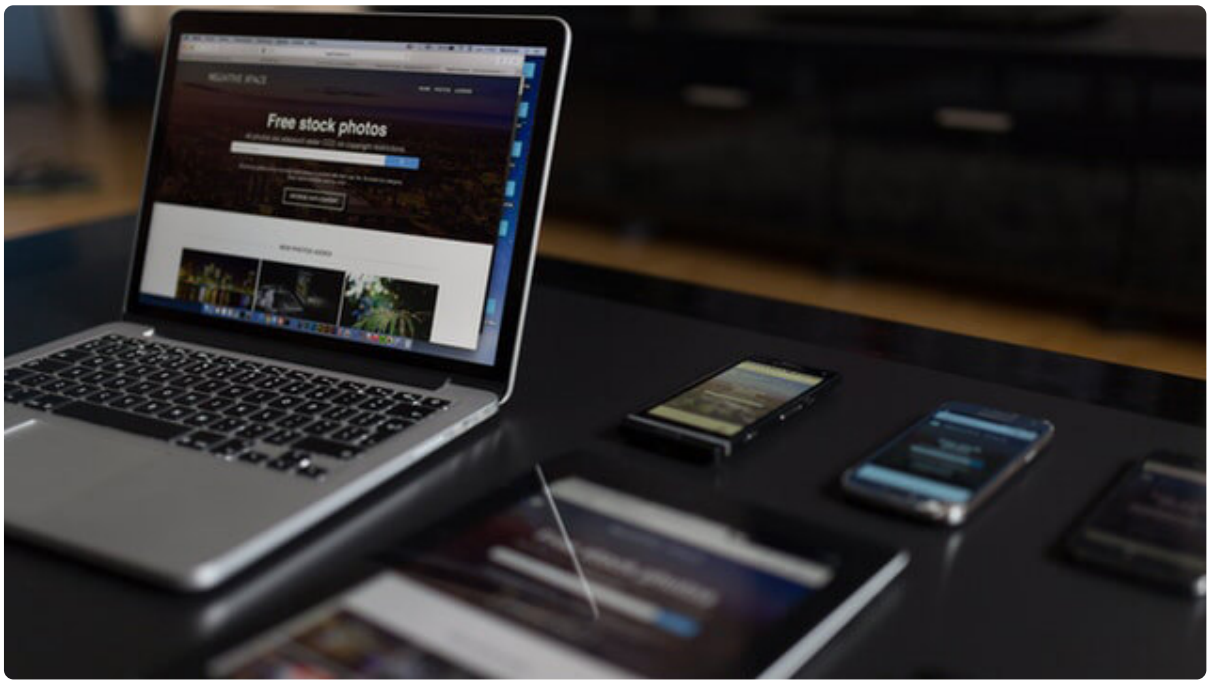
# 21 开发实现积分体系页面

更新时间：2019-08-13 17:02:24




> 老骥伏枥，志在千里； 烈士暮年，壮心不已。
>
> ——曹操

上一节我们已经实现了从微信运动步数获取积分，本节我们将根据第二节的功能设计完成积分体系的页面开发，向用户展示他的积分变动记录。

在第二节中，我们已经使用“分类拆解法”的拆解步骤，将页面进行了详细拆解。本节我们将按照“分类拆解法”的编程步骤，一步一步完成如图 4 所示页面的开发。

**图 4 积分体系页面**


请先完成本章第三节的实践作业，重新获取自己的微信运动步数信息，否则在积分变动记录表 `user_point` 中没有自己的积分获取记录。

# 1. 数据服务

在第五章第四节中我们已经介绍了分层模型，把业务数据访问与前端业务逻辑分层实现，将访问云数据库的操作单独编写在小程序客户端的一系列 Data Service 函数中。

因此，在实现积分体系页面前，需要先准备好积分变动记录表 `user_point` 的 Data Service 函数 PointService.js 。

PointService.js 中主要包括一个类 PointService ，在该类中提供了一个方法 `getPointChangeList` 用于从云数据库的 `user_point` 表中**分页获取**用户的积分变动记录，该方法会调用入参异步回调函数 `success_callback(pointChangeArray)` ，将积分变动记录交给异步回调函数进行后续业务处理。

为了满足积分体系页面对不同类型积分查询的需求， `getPointChangeList` 还需要两个输入参数：

- `type` 参数表示需要获取哪种类型的积分记录，增加积分记录的参数值为 inc ，减少积分记录的参数值为 dec ，全部积分记录的参数值为 all；
- `isReset` 参数表示是否需要重置分页，从第一页开始重新查询积分记录，该参数在用户切换积分记录的查询类 `type` 时为 `true` ，否则为 `false` 。

**分页获取列表数据的具体实现方式在第三章第二节的 “4. 列表数据显示” 中有详细讲解，具体算法思路请回顾该节内容。**

完整的 PointService.js 代码内容如下：

```
// 初始化 云数据库
const db = wx.cloud.database()
const _ = db.command

//引用时间格式化工具
const util = require('../utils/util.js')

/**
 *积分数据操作类
 *@class PointService
 *@constructor
 */
class PointService {
  /**
   * 构造函数
   */
  constructor() {
    this.listIndex = 0 //分页获取数据的当前页索引
    this.pageCount = 12 //每次分页获取多少数据
  }

  /**
   * 从数据库获取用于显示的积分变动列表信息
   * @method getPointChangeList
   * @for PointService
   * @param {string} type 需要获取的积分变动类型，增加-inc，减少-dec，全部-all
   * @param {bool} isReset 是否清空分页缓存
   * @param {function} successCallback(pointChangeArray) 处理数据查询结果的回调函数
   * pointChangeArray数据结构：
   * [{
   *    index,
   *    date,
   *    changePoints,
   *    operation
   * },]
   */
  getPointChangeList(type, isReset, successCallback) {
    var pointChangeArray = []

    //如果传入的积分变动类型为空，直接返回
    if (type === undefined) {
```

```
  typeof successCallback == "function" && successCallback(pointChangeArray)
  return
}

//重置分页为初始值，从第一页开始重新查询积分记录
if (isReset) {
  this.listIndex = 0
  this.pageCount = 12
}

//根据 type 构造查询条件
//默认为查询全部积分记录
var query = db.collection('user_point')
//如果 type 为 inc，只查询增加积分记录
if (type == "inc") {
  query = query
    .where({
      changePoints: _.gte(0)
    })
//如果 type 为 dec，只查询消耗积分记录
} else if (type == "dec") {
  query = query
    .where({
      changePoints: _.lt(0)
    })
}

//按照第三章第二节的 “4. 列表数据显示” 中的算法构造分页查询条件
var offset = this.listIndex * this.pageCount
//skip和limit的传入参数必须大于0
if (offset === 0) {
  query = query
    .orderBy('date', 'desc') //按积分变动时间倒序排序
    .limit(this.pageCount) //限制返回数量为 max 条
} else {
  query = query
    .orderBy('date', 'desc') //按积分变动时间倒序排序
    .skip(offset) // 跳过结果集中的前 offset 条，从第 offset + 1 条开始返回
    .limit(this.pageCount) //限制返回数量为 max 条
}

//按照第三章第二节的 “4. 列表数据显示” 中的算法进行分页查询
var that = this
//执行数据库查询
query
  .get()
  .then(res => {
    if (res.data.length > 0) {
      for (var i in res.data) {
        //由于数据库中存储的 date 类型数据直接显示在页面中会让用户难以理解，
        //需要调整时间的显示格式
        pointChangeArray.push({
          index: res.data[i]._id,
          date: util.formatTime(res.data[i].date),
          changePoints: res.data[i].changePoints,
          operation: res.data[i].operation
        })
      }
      //分页显示的页码+1
      ++that.listIndex
    }
    //回调函数处理数据查询结果。按照第三章第二节的 “4. 列表数据显示” 中的算法，
    //回调函数需要将新获取到的数据添加到数据集合末尾，并设置数据全部加载完毕的标志
    typeof successCallback == "function" && successCallback(pointChangeArray)
  })
  .catch(err => {
    //访问数据库失败 的 系统异常处理
    //跳转出错页面
    wx.redirectTo({
      url: '../../errorpage/errorpage'
    })
    console.error(err)
  })
```

```
  }
}

export default PointService
```

**贴出完整代码，是想向各位同学再次重申代码编写规范性和代码注释完整性的重要性。**

**再次推荐各位同学认真学习：腾讯alloyteam团队前端代码规范。**

## 2. 编程步骤

在编写完积分变动记录表 `user_point` 的数据服务后，我们就可以正式开始页面的编写。

#### 2.1 定义页面子部件及其排列顺序

在本章第二节我们已经完成了图 4 的页面拆解工作。页面可以拆解为 2 个子部件：

- 子部件 1 ：顶部的积分记录类型菜单
- 子部件 2 ：积分记录列表

首先在 WXML 页面模板中定义 2 个子部件的容器，积分体系页面的整体结构可以使用 WeUI 的组件 Navbar 实现：

```
<!-- weui的navbar -->
<view class="weui-tab">
  <!-- 子部件 1 ：顶部的积分记录类型菜单 navbar的选项卡 -->
  <view class="weui-navbar">
  </view>
  <!-- 子部件 2 ：积分记录列表 navbar的内容区域 -->
  <view class="weui-tab__panel">
  </view>
</view>
```

#### 2.2 实现子部件 1 ：顶部的积分记录类型菜单

在本章第二节已经将子部件 1 拆解为一系列的显示元素：

“全部”菜单，显示全部积分记录

单条内容交互界面，事件为用户点击屏幕事件，数据为用户当前选中的菜单 ID。

"增加"菜单，显示积分获取记录

单条内容交互界面，事件为用户点击屏幕事件，数据为用户当前选中的菜单 ID。

"减少"菜单，显示积分消耗记录

单条内容交互界面，事件为用户点击屏幕事件，数据为用户当前选中的菜单 ID。

三个菜单的用户点击屏幕事件类似，事件响应为：

- 设置用户当前选中的菜单 ID 数据 为用户点击的菜单 ID。
- 用户当前选中的菜单 ID 对应菜单的单条内容交互界面为绿色选中样式，其它菜单的界面为未选中样式。
- 从积分变动记录表中获取用户当前选中的菜单 ID 对应类型的积分变动记录第一个分页数据，并存放到积分变动记录数组中。

此外，还有一个系统自动执行的事件：页面加载时默认选中“全部”菜单，然后从积分变动记录表中获取该菜单对应类型的积分变动记录第一个分页数据，并存放到积分变动记录数组中。

#### 2.2.1 实现 Navbar

参照 WeUI 的 Demo 小程序中组件 Navbar 的代码，首先在 data 中定义 Navbar 组件的数据，并设置菜单的名称和菜单对应的 PointService 的查询类型 queryParam：

```
data: {
  tabs: ["全部", "增加", "减少"], //定义navbar的选项卡
  queryParam: ["all", "inc", "dec"], //对应选项卡的查询参数
  activeIndex: 0, //navbar的当前选中选项，页面加载时默认选中“全部”菜单
  sliderOffset: 0, //weui的navbar参数
  sliderLeft: 0, //weui的navbar参数
},
```

然后在 WXML 页面文件中复制组件 Navbar 的页面代码，并绑定菜单点击事件响应函数 tabClick：

```
<!-- navbar的选项卡 -->
<view class="weui-navbar">
  <block wx:for="{{tabs}}" wx:key="*this">
    <view id="{{index}}" class="weui-navbar__item {{activeIndex == index ? 'weui-bar__item_on' : ''}}" bindtap="tabClick">
      <view class="weui-navbar__title">{{item}}</view>
    </view>
  </block>
  <view class="weui-navbar__slider" style="left: {{sliderLeft}}px; transform: translateX({{sliderOffset}}px); -webkit-transform: translateX({{sliderOffset}}p
x);">  </view>
</view>
```

最后在页面加载事件 onLoad 中初始化 Navbar 组件：

```
/**
 * 生命周期函数--监听页面加载
 */
onLoad: function(options) {
  var that = this;
  //初始化weui的navbar
  wx.getSystemInfo({
    success: function(res) {
      that.setData({
        sliderLeft: (res.windowWidth / that.data.tabs.length - sliderWidth) / 2,
        sliderOffset: res.windowWidth / that.data.tabs.length * that.data.activeIndex
      });
    }
  });
},
```

#### 2.2.2 实现事件

页面加载时和用户点击菜单时，事件响应函数都需要调用 PointService 的方法 getPointChangeList 获取用户的分页积分记录。

根据第三章第二节的 “4. 列表数据显示” 中介绍的算法，需要定义保存从数据库中获取到的数据的数组 pointChangeRecords 和是否云开发数据库中所有数据都已经显示到页面中的标志 isNoMoreData：

```
data: {
  pointChangeRecords: [], //积分变化记录
  isNoMoreData: false, //记录是否已加载完所有分页数据
},
```

然后调用 PointService 的方法 `getPointChangeList` 获取积分记录，并根据第三章第二节的“4. 列表数据显示”中介绍的算法进行分页加载数据的逻辑实现：

```
//引用积分变动记录的数据库访问类
import PointService from '../../../../../dataservice/PointService.js'
var pointService = new PointService()

/**
 * 调用 PointService 的 getPointChangeList 方法获取积分记录
 */
getPointList(isTabChanged) {
 //如果用户点击菜单切换了积分记录的查询类型，清空原积分记录数组，从第一页开始重新查询积分记录
 if (isTabChanged) {
  this.setData({
   pointChangeRecords: [],
   isNoMoreData: false
  })
 }

 var pointChangeRecords = this.data.pointChangeRecords
 var that = this
 //调用 PointService 的 getPointChangeList 方法
 pointService.getPointChangeList(
  this.data.queryParam[this.data.activeIndex], //用户当前选中的菜单对应的 type 类型
  isTabChanged, //是否用户点击菜单切换了积分记录的查询类型
  //处理数据库查询结果的回调函数
  function(pointChangeArray) {
   //判断本次查询是否没有获取到数据
   if (pointChangeArray.length > 0) {
    //使用数组的 concat 方法将新获取到的数据添加到数据集合末尾
    pointChangeRecords = pointChangeRecords.concat(pointChangeArray)
    //更新数据集合，使页面显示新获取到的数据
    that.setData({
     pointChangeRecords: pointChangeRecords
    })
   } else {
    //如果查询没有获取到数据，说明云开发数据库中所有数据都已经获取完毕
    //设置数据全部加载完毕的标志
    that.setData({
     isNoMoreData: true
    })
   }
  })
},
```

最后分别实现事件响应函数：

页面加载完毕时系统自动执行的事件

页面加载时默认选中“全部”菜单，页面加载完毕后获取全部积分记录的第一页分页数据：

```
/**
 * 生命周期函数--监听页面初次渲染完成
 */
onReady: function() {
 this.getPointList(true) //页面加载完毕后获取全部积分记录的第一页分页数据
},
```

用户点击菜单事件

当用户单击的菜单不是当前选中菜单时，获取用户新选择菜单的积分记录的第一页分页数据：

```
/**
 * Tab点击事件
 */
tabClick: function(e) {
  //当用户单击的菜单不是当前选中菜单时,才重新设置及选中菜单并重新获取积分记录
  if (e.currentTarget.id != this.data.activeIndex) {
    //重新设置用户选中的菜单为用户点击菜单
    this.setData({
      sliderOffset: e.currentTarget.offsetLeft,
      activeIndex: e.currentTarget.id //记录navbar的当前选中选项
    });
    //获取用户新选择菜单的积分记录的第一页分页数据
    this.getPointList(true)
  }
},
```

**2.3 实现子部件 2 ：积分记录列表**

在本章第二节已经将子部件 2 拆解为一系列的显示元素：

积分记录列表

多条内容交互界面，事件为用户下滑屏幕到页面底部，数据为积分变动记录数组。

积分变动记录数组的第一个分页数据由子部件 1 的事件响应函数设置。

用户下滑屏幕到页面底部事件的事件响应为：从积分变动记录表中分页获取下一页数据，并追加到积分变动记录数组末尾。如果积分变动记录表中所有数据都获取完毕，用户下滑屏幕将不再获取分页数据。

多条内容交互界面显示积分变动记录数组中的所有数据。

积分变化记录数据 `pointChangeRecords` 的显示使用 WXML 的列表渲染语法 `wx:for` ，使用 WeUI 的 Panel 组件作为列表容器，使用WeUI 的 List 组件作为列表显示样式， 使用WeUI 的 Flex 组件实现一条积分记录的左右排列内容显示样式：

```
<!-- 子部件 2 ：积分记录列表 navbar的内容区域 -->
<view class="weui-tab__panel">
 <!-- 使用 Panel 作为积分记录列表的容器 -->
 <view class="weui-panel point_change">
  <view class="weui-panel__bd">
 <!-- 使用 List 作为列表显示样式 -->
   <view class="weui-cells weui-cells_after-title">
    <!-- 列表渲染 列表显示积分详情记录 -->
    <block wx:for="{{pointChangeRecords}}" wx:key="{{pointChange.index}}" wx:for-item="pointChange">
     <!-- 每一条积分记录是一个 cell -->
     <view class="weui-cell weui-cell_hide_line">
      <view class="weui-cell__bd">
       <!-- 一条积分记录的左右排列内容显示样式用 flex 实现 -->
       <view class="weui-flex">
        <!-- 积分变动原因与时间 -->
        <view class="weui-flex__item">
         <view>{{pointChange.operation}}</view>
         <view class="text-sm">{{pointChange.date}}</view>
        </view>
        <!-- 积分变动数值 -->
        <view class="align-center text-xxl">
         <!-- 如果是增加积分用红色显示，如果是消耗积分用绿色显示 -->
         <view class="{{ pointChange.changePoints>0 ? 'text-red' : 'text-green' }}">
          {{pointChange.changePoints>0 ? "+" + pointChange.changePoints : pointChange.changePoints }}
         </view>
        </view>
       </view>
      </view>
     </view>
    </block>
   </view>
  </view>
 </view>
</view>
```

然后按照第三章第二节的 “4. 列表数据显示” 中介绍的方法，定义用户下滑屏幕到页面底部事件的事件响应：

```
/**
 * 用户下滑屏幕到页面底部事件的处理函数
 */
onReachBottom: function() {
 if (!this.data.isNoMoreData) { //如果还有数据未加载完，则获取更多数据
  this.getPointList(false)
 }
},
```

由于篇幅所限，完整的 **WXML**页面模板代码请查阅专栏源代码 **point.wxml** 与 **point.wxss** ， 完整的 **JS** 逻辑代码见 **point.js** ，具体代码位置见本节末尾图 **5**。

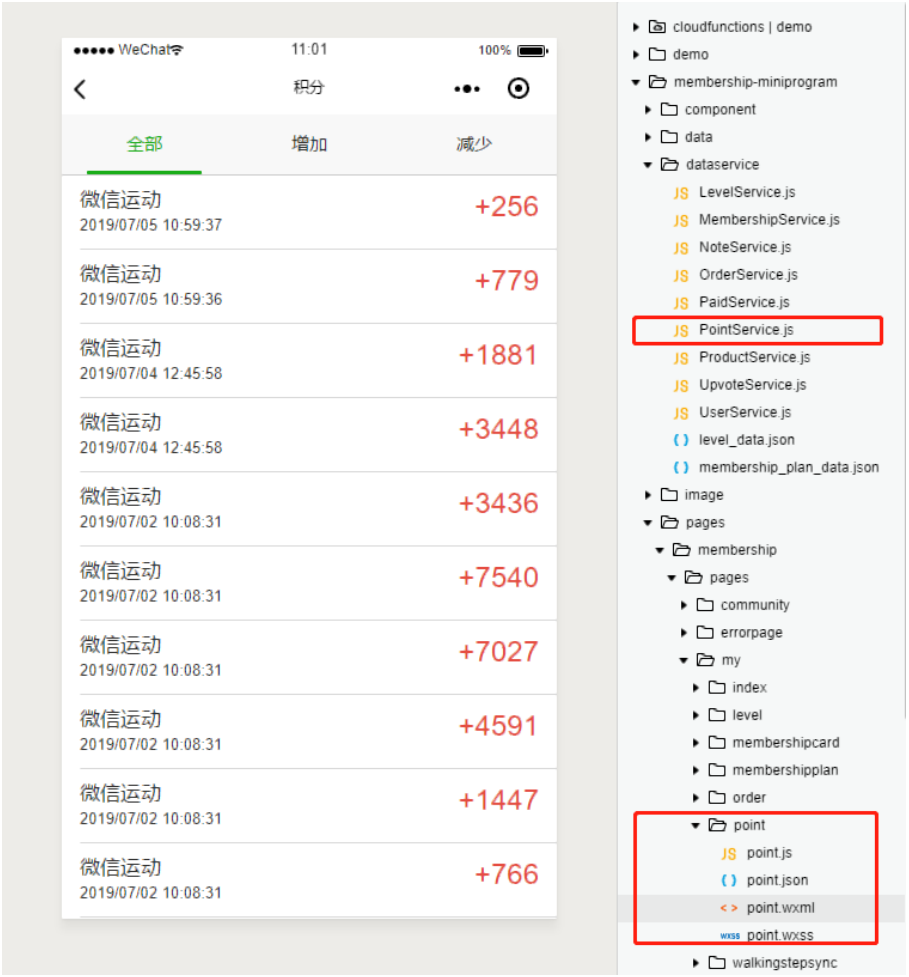
## 3. 专栏源代码

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 5 红框标出的位置。

图 **5** 本节源代码位置




## 下节预告

下一节，我们将实现积分体系的风控规则，同时介绍用户被锁定后的页面展现内容。

## 实践环节

实践是通往大神之路的唯一捷径。

本节实操内容：

- 编写代码完成图 4 所示的页面，如碰到问题，请阅读本专栏源代码学习如何实现。

}

20 开发实现同步微信运动步数获得积分

22 坏人请走开：开发实现积分体系风控规则