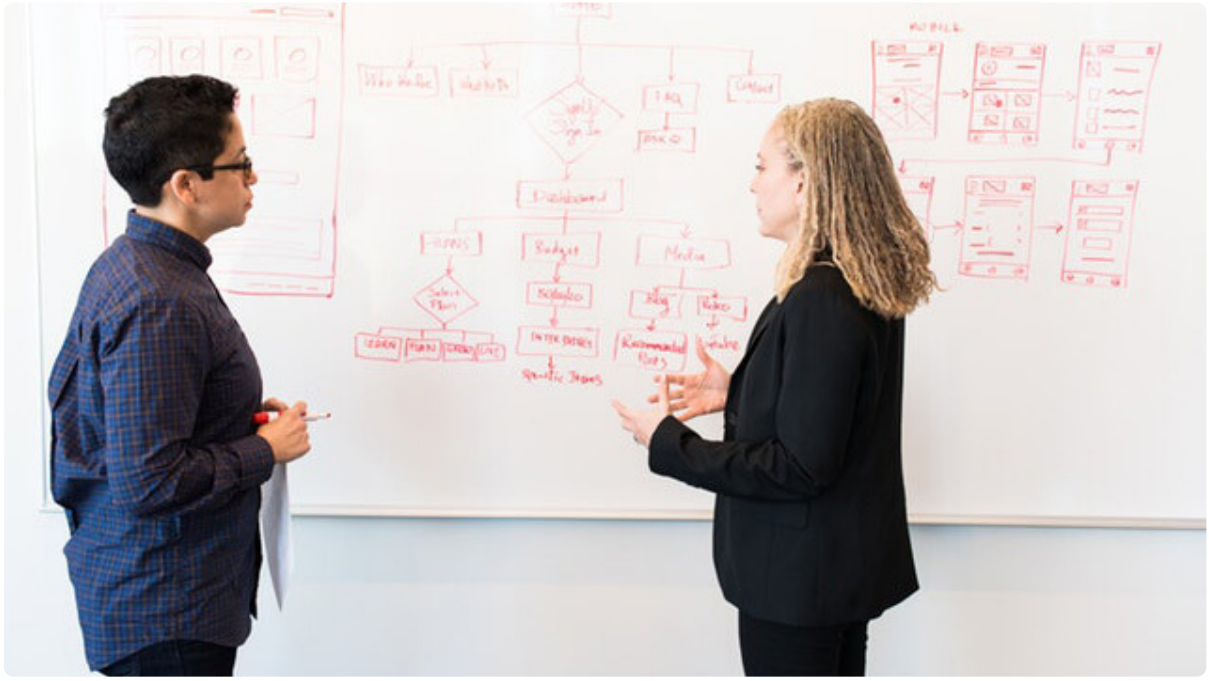
16 开发实现用户成长体系页面

更新时间：2019-07-26 10:58:50

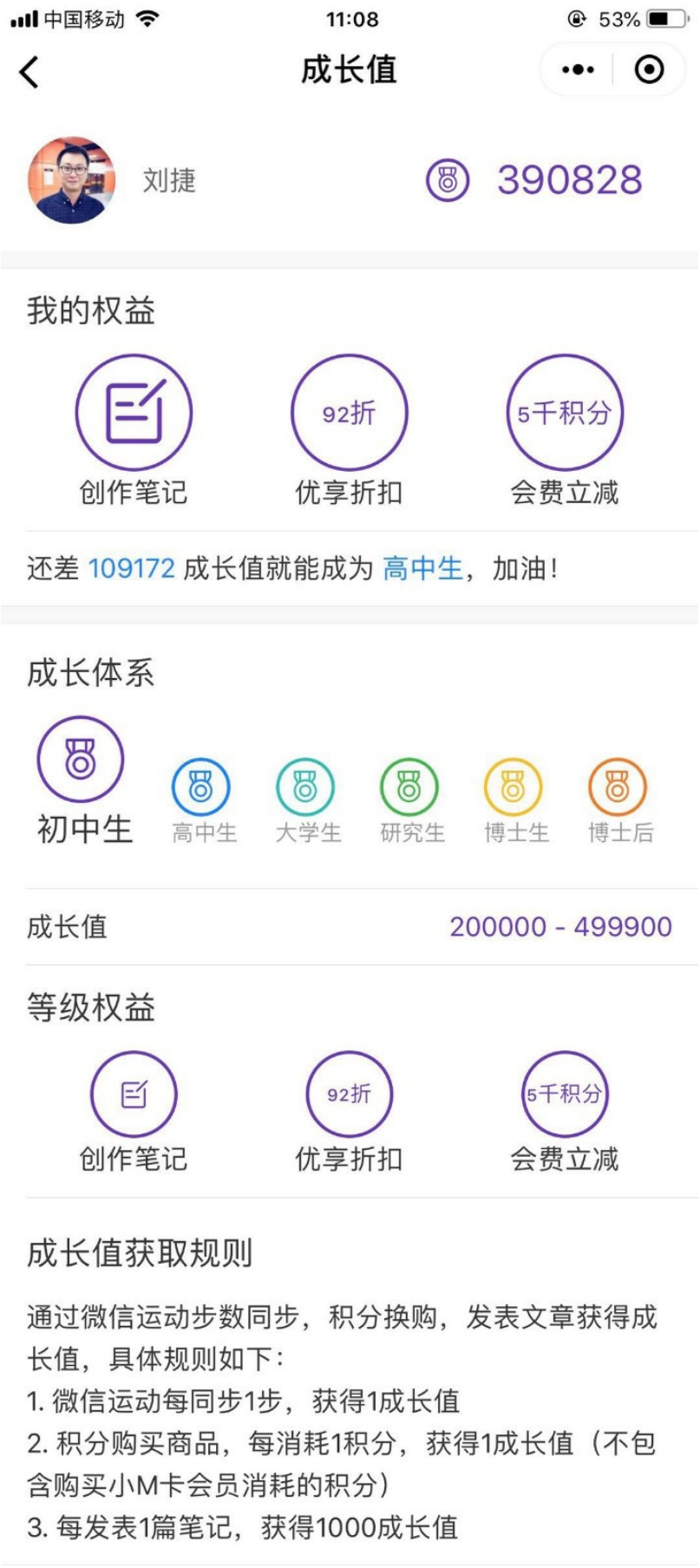


> 世界上最快乐的事，莫过于为理想而奋斗。
>
> ——苏格拉底

上一节我们已经实现了从微信运动步数获取成长值，本节我们将根据第二节的功能设计完成用户成长体系的页面开发，向用户展示他的成长值、等级、特权，并介绍用户成长体系。

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 用户成长体系页面



请先阅读并实现本章第五节中“用户首次打开小程序自动注册”的功能，否则在 `user` 表中没有自己的用户记录，打开本节编写完的页面无法看到自己的成长值、等级、特权等信息。

请先完成本章第三节的实践作业，获取自己的微信运动步数信息，否则在 `user` 表中没有自己的总成长值信息，打开本节编写完的页面无法看到自己的成长值、等级、特权等信息。

1. 数据服务

在实际商业应用开发中，代码结构通常都会使用分层模型。在 Java 开发、.Net 开发中通常使用 DAO 层和 Service 层来为前端的 MVC 提供业务数据访问能力，或者使用 Web API 来为前端的 MVVM 提供业务数据访问能力。

在小程序云开发应用中，我们同样应该考虑使用分层模型，把业务数据访问与前端业务逻辑分层实现。代码分层的最大好处之二是复用和解耦。

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，任何页面要访问数据库都只需要一行语句调用 Data Service 函数即可，不用每次都重新编写数据库访问的代码，这就实现了数据库访问代码的复用。

云函数则是另一种在服务端对数据库访问代码的复用。

至于解耦的好处和前述各种分层设计模式的详细内容，有兴趣的同学可以在网上自行搜索查阅。

因此，在会员制社交电商小程序中，我们将访问每个数据库集合的方法，单独编写在一个 Data Service 目录中，每个数据库集合的访问方法编写在一个单独的 Service.js 文件中，如本节末尾图 5 所示。

本节以 `user` 表的访问方法为例，介绍 UserService.js 的实现。**本节需要用到的另一个数据服务 LevelService.js 请参考专栏源代码，具体代码位置见本节末尾图 5 。**

由于在小程序客户端访问云开发的数据库是使用 ajax 方式，ajax 是异步调用方式，在发出访问云开发数据库请求后，程序将继续执行后续代码，而不是等到云开发数据库返回结果后再继续执行后续代码。

因此我们在 UserService 中需要使用异步回调的方式，才能确保调用数据服务 UserService 的业务逻辑按顺序被正确执行。

不了解 JavaScript 异步机制的同学可以阅读这篇文章详细学习：JS基础——异步回调。

UserService.js 中主要包括一个类 UserService ，在该类中提供了一个方法 `getUserInfo` 用于从云数据库的 `user` 表中获取当前用户的用户信息，在获取到用户信息后，该方法会调用入参异步回调函数 `success_callback(userinfo)` ，将用户信息交给异步回调函数进行后续业务处理。

在访问数据库时，还可能由于网络不稳定等原因导致访问数据库失败等系统异常，需要用 `catch` 捕捉异常并进行异常处理。

此外，业务异常也需要考虑。在用户第一次访问小程序时，系统就自动完成了用户注册，因此在正常情况下一定能查询到该用户的用户信息。如果云数据库的 `user` 表中没有查询到用户信息，则说明出现异常，需要进行异常处理。

完整的 UserService.js 代码内容如下：

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

/**
 *用户信息数据操作类
 *@class UserService
 *@constructor
 */
class UserService {
  /**
   * 构造函数
   */
  constructor() {}

  /**
   * 从数据库获取用户信息
   * @method getUserInfo
   * @for UserService
   * @param {function} success_callback(userinfo) 处理数据查询结果的回调函数
   * userinfo数据结构：
   * {
   *    _id,
   *    _openid,
   *    date, //注册时间
   *    growthValue, //总成长值
   *    point, //可用积分
   *    memberExpDate,  //会员到期时间 Membership Expiration Date
   *    isLocked //是否因触发风控规则被锁定
   * }
   */
  getUserInfo(success_callback) {
    //执行数据库查询
    db.collection('user')
      .get()
      .then(res => {
        if (res.data.length > 0) {
          //回调函数处理数据查询结果
          typeof success_callback == "function" && success_callback(res.data[0])
        } else {
          //没有查询到用户信息 的 业务异常处理
          //跳转出错页面
          wx.redirectTo({
            url: '../../errorpage/errorpage'
          })
        }
      })
      .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 UserService
```

在编程中，代码编写规范性和代码注释完整性是评价代码质量高低非常重要的两个标准，从代码规范性和代码注释中就可以看出写代码的人水平如何。

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 user 表与 level 表的数据服务后，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 3 个子部件，3 个子部件之间由灰色色块区隔。3 个子部件从上到下分别：

- 子部件 1 ：用户的当前用户等级与成长值显示
- 子部件 2 ：用户当前用户等级的特权显示
- 子部件 3 ：用户成长体系介绍

首先在 WXML 页面模板中定义 3 个子部件的容器，并把它们按顺序排列：

```
<!-- 当获取到用户数据后，再显示页面-->
<view wx:if="{{inited}}">
 <!--页面第一部分 用户的当前用户等级与成长值显示 -->
 <view class="padding bg-white">

 </view>

 <!--页面第二部分  用户当前用户等级的特权显示 -->
 <view class="weui-panel">

 </view>

 <!--页面第三部分  用户成长体系介绍 -->
 <view class="weui-panel">

 </view>
</view>
```

WeUI 的 Panel 可以实现子部件之间由灰色色块区隔的效果，同时 Panel 内部的多个显示元素之间会由横线区隔。

另外，使用 ColorUI 丰富的样式布局语法，可以简单地实现设置子部件 1 的背景为白色 `bg-white` 与 内边距 `paddin g` 。

在页面中需要先从数据库读取用户等级与等级特权表 `level` 与用户表 `user` 中用户的当前总积分，经过计算后才能显示用户的等级和用户体系等信息，这需要一定时间。如果从数据库获取数据前就显示页面，用户将先看到页面有很多内容空白，然后再看到数据显示出来，这样的用户体验不太好。

因此在页面中需要增加一个从数据库获取完数据的标志 `inited` ，从数据库获取到数据后再显示页面。

```
data: {
 inited: false, //是否已从数据库读取数据
},
```

2.2 实现子部件 1 ：用户的当前用户等级与成长值显示

在本章第二节已经将子部件 1 拆解为一系列的显示元素：

显示元素 1 ：用户的微信头像

交互界面，无事件，数据为微信用户头像

显示元素 2：用户的微信昵称

交互界面，无事件，数据为微信用户昵称

显示元素 3 ：用户的等级图标

交互界面，无事件，数据为用户当前等级和用户等级图标颜色

用户等级是计算值，根据用户当前成长值，从用户等级与等级特权表（见上一节 4. 完整的用户成长体系）中计算得出

用户等级图标颜色在用户等级与等级特权表中，该表数据需要从云数据库中获取

显示元素 4 ：用户的当前成长值

交互界面，无事件，数据为用户当前成长值，该数据需要从云数据库中获取

用户的当前成长值是成长值获取记录表中每一条记录中的成长值之和

2.2.1 定义显示元素的排列顺序

子部件 1 的显示元素是水平排列，可以使用 WeUI 的 Flex 组件实现水平排列效果。

```
<view class="weui-flex align-center">
</view>
```

2.2.2 实现显示元素 1 ：用户的微信头像与显示元素 2 ：用户的微信昵称

用户的微信头像和微信昵称可以直接调用小程序的开放能力 open-data 进行显示。

open-data 的详细介绍请阅读微信官方“小程序开发文档”的 组件->开放能力->open-data。

```
<!-- 使用open-data显示用户头像 -->
<open-data class="userAvatarUrl" type="userAvatarUrl"></open-data>
<!-- 使用open-data显示用户昵称 -->
<open-data type="userNickName"></open-data>
```

2.2.3 实现显示元素 3 ：用户的等级图标

在显示元素 3 中含有 3 个数据：

- 用户信息，用户信息中包含用户的当前总成长值
- 用户等级与等级特权表，用于计算用户的等级
- 用户等级信息，根据用户的当前总成长值和用户等级与等级特权表计算得出。用户等级信息中包含了用户等级图标颜色

```
data: {
  levels: [], //用户等级与等级特权表
  myLevel: {},  //用户等级信息
  myInfo: {}, //用户信息
},
```

在页面加载时，我们需要调用 `LevelService` 和 `UserService` 获得用户信息及用户等级与等级特权表，同时计算出用户的等级信息。

数据服务需要先在 JS 逻辑中引用后才能使用，语法如下：

```
//引用用户等级与等级特权表的数据库访问类
import LevelService from '../../../../../dataservice/LevelService.js'
var levelService = new LevelService()
//引用用户信息的数据库访问类
import UserService from '../../../../../dataservice/UserService.js'
var userService = new UserService()
```

在引用数据服务后，即可在页面加载时从数据库读取数据，并计算用户等级：

```
/**
 * 生命周期函数--监听页面加载
 */
onLoad: function(options) {
  var that = this
  //先获取用户等级与等级特权表
  levelService.getLevelList(
    //--------------获取用户等级与等级特权表的回调函数
    function(levelList) {
      var levels = levelList
      //再获取用户信息
      userService.getUserInfo(
        //----------------------获取用户信息回调函数
        function(userinfo) {
          var myInfo = userinfo
          that.setLevelData(levels, myInfo)
        }
        //----------------------获取用户信息回调函数
      )
    }
    //--------------获取用户等级与等级特权表的回调函数
  )
},

/**
 * 计算用户等级信息
 */
setLevelData: function(levels, myInfo) {
  var that = this
  //每个用户等级都有成长值上限和下限，用户的当前总成长值在哪个用户等级的上下限区间内，用户就是该等级
  var myLevel = levels.filter(e => e.minGrowthValue <= myInfo.growthValue && myInfo.growthValue <= e.maxGrowthValue )[0]
  //设置数据用于显示
  that.setData({
    levels: levels,
    myLevel: myLevel,
    myInfo: myInfo
  })
  //在获取完数据后，显示界面
  that.setData({
    inited: true
  })
},
```

要同时从数据库获取用户信息及用户等级与等级特权表，需要在第一个数据服务的回调函数中调用第二个数据服务，请注意多个数据服务调用的嵌套写法。

熟悉 Promise 的同学可以尝试使用 Promise 的语法来避免回调地狱。

最后，使用 ColorUI 的图标组件，根据用户的等级，显示用户等级对应的图标颜色：

```
<!-- 显示用户等级对应的图标颜色 -->
<view class="line-{{ myLevel.icon }}">
  <text class="icon-medal"> </text>
</view>
```

2.2.4 实现显示元素 4：用户的当前成长值

显示元素 4 的数据在 2.2.3 已经得到，直接显示出来，并用 ColorUI 的颜色样式显示用户等级对应的颜色即可：

```
<!-- 显示用户当前总成长值 -->
<view class="line-{{ myLevel.icon }}">
  <text> {{myInfo.growthValue}} </text>
</view>
```

由于篇幅所限，包含完整样式的子部件 **1 WXML**页面模板代码请查阅专栏源代码 **level.wxml** 与 **level.wxss**，完整的 **JS** 逻辑代码见 **level.js**，具体代码位置见本节末尾图 **5** 。

2.3 实现子部件 **2**：用户当前用户等级的特权显示

在本章第二节已经将子部件 2 拆解为一系列的显示元素：

显示元素 1：标题 我的权益**（代码参见专栏源代码）**

静态界面，无事件，无数据

显示元素 2：用户当前等级特权内容显示

交互界面，无事件，数据为用户当前等级对应的特权内容

特权内容在用户等级与等级特权表中，该表数据需要从云数据库中获取

显示元素 3：用户等级提升还需要多少成长值

交互界面，无事件，数据为用户升级所需成长值

用户升级所需成长值是计算值，由用户下一等级最低所需成长值 - 用户当前成长值计算得出

用户下一等级最低所需成长值在用户等级与等级特权表中，该表数据需要从云数据库中获取

2.3.1 实现显示元素 2：用户当前等级特权内容显示

显示元素 4 的数据在 2.2.3 已经得到。

特权的水平排列效果使用 WeUI 的 Flex 组件实现。用 ColorUI 的头像元素与图标组件实现特权的镂空图标显示效果。用 ColorUI 的颜色样式设置用户等级对应的特权颜色。

特权有多个，需要使用 WXML 的列表渲染。

创作笔记的特权描述是图标，其它特权的特权描述是显示特权描述内容，需要分别进行显示。

```
<view class="weui-flex text-center">
  <!-- 列表渲染 -->
 <block wx:for="{{myLevel.bonus}}" wx:key="this">
    <!-- 使用 WeUI 的 Flex 组件实现特权横向排列 -->
  <view class="weui-flex__item">
     <!-- 特权描述 -->
 <view class="cu-avatar xl round bg-white line-{{myLevel.icon}} solids">
       <!-- 其他特权 -->
  <text wx-if="{{item.desc.length > 0}}" class="avatar-text">{{item.desc}}</text>
  <!-- 创作笔记特权 -->
      <view wx-if="{{item.desc.length <= 0}}" class="text-sl">
  <text class="icon-edit"></text>
  </view>
 </view>
     <!-- 特权名称 -->
 <view>{{item.name}}</view>
  </view>
 </block>
</view>
```

2.3.2 实现显示元素 3：用户等级提升还需要多少成长值

在显示元素 3 中含有 2 个新数据：

- 用户下一等级信息
- 用户升级所需成长值，由用户下一等级最低所需成长值 - 用户当前成长值计算得出

```
data: {
  nextLevel: undefined, //用户当前等级的下一等级信息
  growthValueToNextLevel: -1, //用户还需要多长成长值才能升到下一级
},
```

计算过程需要添加在 2.2.3 的 setLevelData 函数中：

```
//获取下一用户等级
var nextLevel = levels.filter(e => e.id == myLevel.id + 1)[0]
//如果用户没有升级到最高级，就设置用户的下一等级信息
if (nextLevel !== undefined) {
  that.setData({
    growthValueToNextLevel: nextLevel.minGrowthValue - myInfo.growthValue,
    nextLevel: nextLevel
  })
}
```

最后将数据显示到页面中：

```
<!-- 如果用户已经升级到最高等级，则不应该再显示用户的下一等级信息 -->
<view wx:if="{{ growthValueToNextLevel > 0 }}" class="weui-cell weui-cell_hide_line">
  <view class="weui-cell__bd">
    <view>
      还差
      <text class="line-{{nextLevel.icon}}"> {{growthValueToNextLevel}} </text>成长值就能成为
      <text class="line-{{nextLevel.icon}}"> {{nextLevel.title}}</text>，加油！
    </view>
  </view>
</view>
```

由于篇幅所限，包含完整样式的子部件 2 WXML页面模板代码请查阅专栏源代码 level.wxml 与 level.wxss，完整的 JS 逻辑代码见 level.js，具体代码位置见本节末尾图 5 。

2.4 实现子部件 3：用户成长体系介绍

在本章第二节已经将子部件 3 拆解为一系列的显示元素：

显示元素 1：标题 成长体系**（代码参见专栏源代码）**

静态界面，无事件，无数据。

显示元素 2：所有用户等级的水平列表显示

交互界面，事件为用户屏幕滑动事件与用户点击屏幕事件，数据为所有用户等级的名称与等级图标。

用户屏幕滑动事件的事件响应为：根据用户向左或向右滑动屏幕的方向，显示更低或更高用户等级的名称与图标。

用户点击屏幕事件的事件响应为：将用户点击选中的用户等级图标和文字放大显示，其余用户等级正常显示，同时修改第三个显示元素的数据为用户点击的用户等级的成长值范围，修改第四个显示元素的数据为用户点击的用户等级的等级特权内容。

所有用户等级的名称与等级图标在用户等级与等级特权表中，该表数据需要从云数据库中获取。

显示元素 3：显示用户当前选中的用户等级的成长值范围

交互界面，无事件，数据为用户当前选中的用户等级对应的成长值范围，该数据内容由第二个显示元素的用户点击屏幕事件决定。

显示元素 4 ：显示用户当前选中的用户等级的等级特权内容

交互界面，无事件，数据为用户前选中的用户等级对应的特权内容，该数据内容由第二个显示元素的用户点击屏幕事件决定。

显示元素 5 ：显示成长值获取规则说明**（代码参见专栏源代码）**

静态界面，无事件，无数据。

成长值获取规则说明内容可直接写在 WXML 页面模板中。

2.4.1 实现显示元素 2 ：所有用户等级的水平列表显示

显示元素 2 的事件需要根据用户的点击行为切换选中的用户等级并在显示元素 3 和显示元素 4 中显示选中用户等级的详细信息，因此需要两个数据来记录当前选中的用户等级：

```
data: {
 selectedLevel: {}, //用户当前选中的成长等级信息
 selectedId: 1, //用户当前选中的成长等级id
},
```

同时，需要在 2.2.3 的 `setLevelData` 函数中设置默认选中的用户等级为用户的当前等级：

```
//设置默认选中的用户等级为用户的当前等级
that.setData({
   selectedId: myLevel.id,
   selectedLevel: levels[myLevel.id - 1]
})
```

然后定义用户点击用户等级的事件函数：

```
/**
 * 用户点击用户等级事件
 */
onLevelItemClick: function(event) {
   var id = event.currentTarget.dataset.item.id;
   //如果用户点击的用户等级与当前选中的用户等级不一致，则切换当前选中的用户等级
   if (this.data.selectedId != id) {
      this.setData({
         selectedId: id,
         selectedLevel: this.data.levels[id - 1]
      });
   }
},
```

最后在页面中实现所有用户等级的横向滚动，并绑定用户点击事件的处理函数 `onLevelItemClick`。

在小程序的官方组件中提供了 scroll-view 组件，该组件支持横向和纵向滚动。

scroll-view 的详细介绍请阅读微信官方“小程序开发文档”的 组件->视图容器->scroll-view，在小程序组件 Demo 中也有 scroll-view 的示例（Demo 位置见第三章第一节）。**需要注意的是，横向滚动需要设置样式** `white-space: nowrap;` 。

用户选中的用户等级，用户等级图标和用户等级名称需要放大显示，这可以使用 ColorUI 的样式布局语法实现。

```
<!-- 使用 scroll-view 实现横向滚动，横向滚动需要设置样式 white-space: nowrap; -->
<!-- 设置 scroll-into-view属性 实现在用户打开页面时自动滚动到用户的当前等级 -->
<scroll-view scroll-x="true" scroll-into-view="{{myLevel.icon}}" class="level_list_container">
  <!-- 列表渲染 -->
  <block wx:for="{{levels}}" wx:for-item="level" wx:key="{{level.id}}">
    <!-- 绑定用户点击事件 -->
    <view id="{{level.icon}}" class="level_tab_item margin-sm" bindtap="onLevelItemClick" data-item="{{level}}">
      <!-- 当前选中的用户等级图标放大显示 -->
      <view class="cu-avatar {{ level.id == selectedId ? 'lg' : '' }} round bg-white line-{{ level.icon }} solids">
        <text class="icon-medal"></text>
      </view>
      <!-- 当前选中的用户等级名称放大显示 -->
      <view class="{{ level.id == selectedId ? 'text-xl text-black' : '' }}">{{level.title}}</view>
    </view>
  </block>
</scroll-view>
```

2.4.2 实现显示元素 3：显示用户当前选中的用户等级的成长值范围

显示元素 3 的实现较为简单，左右水平排列使用 WeUI 的 Flex 组件实现。

需要注意的是，最高用户等级的成长值范围需要特殊处理，最高用户等级的成长值范围应该显示为 "10000000+ "，而不应该显示为 “10000000 - -1”（-1 为 最高用户等级 “博士后” 定义的 `maxGrowthValue` 值）。

```
<!-- 用户选中成长等级的成长值范围 -->
<view class="weui-cell weui-cell_hide_line">
  <view class="weui-cell__bd">
    <view class="weui-flex">
      <view class="weui-flex__item">成长值</view>
      <view>
  <!-- 如果不是最高等级显示 XXX - YYY的形式，如果是最高等级显示 ZZZ+ 的形式 -->
        <text class="line-{{selectedLevel.icon}}">{{ selectedLevel.minGrowthValue }}{{ selectedLevel.maxGrowthValue >0 ? ' - ' + selectedLevel.maxGrowthValue : '+' }}</text>
      </view>
    </view>
  </view>
```

2.4.3 实现显示元素 4：显示用户当前选中的用户等级的等级特权内容

显示元素 4 的实现方式与 2.3.1 一致，唯一不同的是显示数据是 `selectedLevel` 而不是 `myLevel` 。实现代码请参考 2.3.1 。

由于篇幅所限，包含完整样式的子部件 3 WXML页面模板代码请查阅专栏源代码 level.wxml 与 level.wxss ，完整的 JS 逻辑代码见 level.js ，具体代码位置见本节末尾图 5 。

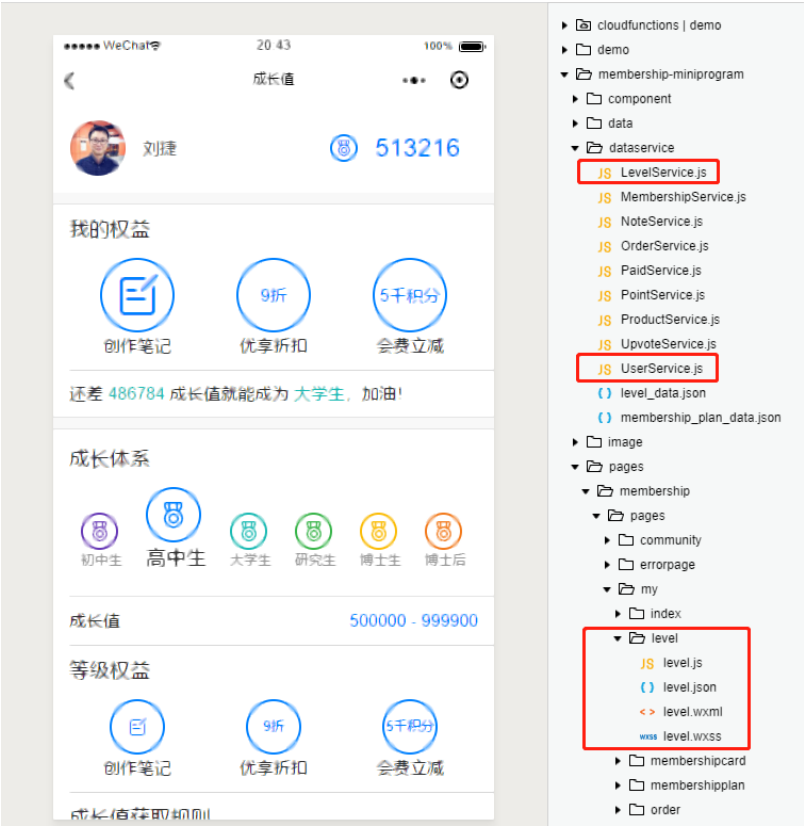
3. 专栏源代码

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，我们将实现用户成长体系的风控规则，同时介绍用户第一次登录时自动注册的实现方式。

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。

15 开发实现同步微信运动步数获得成长值

17 坏人请走开：开发实现用户成长体系风控规则

精选留言 2

欢迎在这里发表留言，作者筛选后可公开显示

qq_慕用4296762

老师，我有个疑惑，我看到UserService.js,获取用户应该是获取当前登录用户，然而我在源码里面看到的是res.data[0]，这个是返回的第一个用户吧，这种写法我在前面章节也看到富哦，也请教过，希望老师帮忙抽空看看，真的很疑惑。 getUserInfo(success_callback) { //执行数据库查询 db.collection('user') .get() .then(res => { if (res.data.length > 0) { //回调函数处理数据查询结果 typeof success_callback == "function" && success_callback(res.data[0])

0 回复 5天前

qq_慕用4296762 回复
qq_慕用4296762

不知道有没有其他同学和我类似疑惑，老师也没时间给回复， 我现在是在全局变量里面定义openid然后在UserService里面判断openid是否是登录的当前用户，我觉得源码里面res.data[0]是不正确的，每个用户并不是单独的存一张表，而是所有用户都存在user表中 var my = res.data.filter(m => m._openid === app.globalData.openid)[0]

回复 2天前

慕九州1482116 回复
qq_慕用4296762

数据库get操作返回的是列表，就算是只有一条结果也是列表，所以要取出第一个元素

回复 2天前

作者 **刘捷Jay** 回复
qq_慕用4296762

这个问题跟你在 08 中的提问是一个问题，答案也是一样的：你的问题在小程序开发文档的 云开发-开发指引-数据库-基础概念-权限管理 中有详细的讲解，具体地址是：https://developers.weixin.qq.com/miniprogram/dev/wxcloud/guide/database/permission.html。 user数据集合的权限是“仅创建者可读写”，因此在小程序客户端只能读到自己的用户信息。每个用户在user数据集合只有一条记录，因此res.data[0]就是当前登录用户的信息。

回复 2天前

点击展开后面 3 条

qq_慕用4296762

说先阅读本章第五节自动注册部分，这次最新的不才第四节么

0 回复 5天前

作者 **刘捷Jay** 回复
qq_慕用4296762

第五节已经发布啦，可以看啦

回复 2天前