40 朋友圈刷起来：开发实现 UGC 社区首页面

更新时间：2019-09-23 10:38:32

> 人生太短，要干的事太多，我要争分夺秒。
>
> ——爱迪生

本节我们将实现 UGC 社区首页，用瀑布流列表的方式显示用户发表的笔记。

在第二节中，我们已经使用“分类拆解法”的拆解步骤，对 UGC 社区首页（如图 9 所示）进行了详细拆解。本节我们将按照“分类拆解法”的编程步骤，一步一步完成页面开发。

图 9 UGC 社区首页

如还未在云数据库中创建点赞记录表 `upvote` 的同学，请先在云数据中新建该表。

1. 数据服务

根据本章第二节的页面拆解结果，UGC 社区首页有 3 个菜单栏，分别为：

- 发现：显示所有笔记列表
- 收藏：显示用户点赞过的笔记列表
- 我的：显示用户自己发表的笔记列表

每个菜单显示的笔记列表均不相同。

发现菜单和我的菜单的数据需要从笔记记录表 `user_note` 中获取，因此我们要在数据服务文件 NoteService.js 的 `NoteService` 类中增加一个公开方法 `getNoteList` ，从笔记记录表 `user_note` 中**分页获取**所有笔记列表和用户自己发表的笔记列表。

`getNoteList` 的整个实现逻辑与第六章第四节的 `getPointChangeList` 基本一致，主要区别是查询条件。笔记查询的条件有两类：

- 第一类是查询所有笔记，按笔记发表时间降序排序
- 第二类是查询用户自己发表的笔记，按笔记发表时间降序排序

因此，`getNoteList` 的查询参数是用户的 openId，如果 openId 为空字符串，则查询所有笔记，否则查询 openId 发表的笔记。

`getNoteList` 中构造查询条件的代码如下：

```
/**
 * 从数据库获取笔记列表信息
 * @method getNoteList
 * @for NoteService
 * @param {string} openId 用户OpenId，为空查询全部用户
 * @param {bool} isReset 是否清空分页缓存
 * @param {function} successCallback(noteArray) 处理数据查询结果的回调函数
 * noteArray数据格式：
 * [{
 *    index,  //笔记ID
 *    date,   //笔记发表时间
 *    content,  //笔记文字内容
 *    img,  //笔记第一张图片地址
 *    height,  //笔记第一张图片高度（px）
 *    width,  //笔记第一张图片宽度（px）
 *    type  //数组类型，笔记为'note'
 * },]
 */
getNoteList(openId, isReset, successCallback) {
  //... 略，详见专栏源代码

  //构造查询条件
  //默认查询条件为获取全部笔记
  var query = db.collection('user_note')
  if (openId != "") {
    //如果 openId 有值，仅查询 openId 发表的笔记
    query = query.where({
      _openid: openId
    })
  }

  //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') //按笔记发表时间降序排序
      .limit(this.pageCount) //限制返回数量为 pageCount 条
  } else {
    query = query
      .orderBy('date', 'desc') //按笔记发表时间降序排序
      .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 pageCount 条
  }

  //... 略，详见专栏源代码
}
```

`getNoteList` 的完整实现请回顾第六章第四节 `getPointChangeList` 方法的代码，或参考专栏源代码 **NoteService.js**，具体代码位置见本节末尾图 **10**。

收藏菜单显示用户点赞过的笔记列表，用户点赞信息需要从点赞记录表 `upvote` 中查询，因此我们要新建数据服务文件 UpvoteService.js，在文件中编写点赞记录表的数据服务类 `UpvoteService`，在类中实现一个公开方法 `getMyUpvoteList`，从从点赞记录表 `upvote` 中**分页获取**用户的点赞记录列表。

`getMyUpvoteList` 中构造查询条件的代码如下：

```
/**
 * 从数据库获取点赞列表信息
 * @method getMyUpvoteList
 * @for UpvoteService
 * @param {bool} isReset 是否清空分页缓存
 * @param {function} successCallback(upvoteArray) 处理数据查询结果的回调函数
 * upvoteArray数据结构：
 * [{
 *    _id,
 *    _openid,
 *    date,
 *    noteId,
 *    autherOpenid
 * },]
 */
getMyUpvoteList(isReset, successCallback) {
  //... 略，详见专栏源代码

  //构造根据点赞者 OpenId 查询的查询条件
  var query = db.collection('upvote')
    .where({
      _openid: app.globalData.openid
    })

  //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') //按点赞时间降序排序
      .limit(this.pageCount) //限制返回数量为 pageCount 条
  } else {
    query = query
      .orderBy('date', 'desc') //按点赞时间降序排序
      .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 pageCount 条
  }

  //... 略，详见专栏源代码
}
```

`getMyUpvoteList` **的完整实现请回顾第六章第四节** `getPointChangeList` **方法的代码，或参考专栏源代码** **`UpvoteService.js`** **，具体代码位置见本节末尾图 10。**

由于点赞记录表中只保存了笔记 ID，笔记的信息需要从笔记记录表 `user_note` 中获取。

获取用户点赞过的笔记列表的算法思路与第八章第八节获取订单列表的算法类似：

- 第一步，调用数据服务获取一页点赞记录列表；
- 第二步，将点赞记录列表中每个笔记 ID 都存放到一个全量笔记 ID 数组中，存放时去重；
- 第三步，在笔记信息的数据服务中增加一个方法 `getNotesByIndex`，输入参数为全量笔记 ID 数组，返回结果为全量笔记 ID 数组对应的全量笔记信息数组，该方法仅请求一次数据库。

在第三步得到的全量笔记信息数组即为收藏菜单显示的用户点赞过的笔记列表。

该算法需要在 `NoteService` 类中增加一个方法 `getNotesByIndex` 。 `getNotesByIndex` **的具体实现代码请阅读专栏源代码的** `NoteService.js` **文件。**

2. 编程步骤

在编写好数据服务后，就可以根据拆解结果一步步完成页面的代码编写。

本节讲解的 UGC 社区首页实现方式是直接在页面中实现所有功能，在源代码中使用了自定义组件的方式实现子部件 2 。

瀑布流笔记列表的自定义组件源代码位于 `\membership-miniprogram\component\WaterFallView\` 目录中。

学有余力的同学在按照本节讲解的方式实现 UGC 社区首页后，可以继续学习微信官方“小程序开发文档”中自定义组件的内容（文档位置：“指南” -> “自定义组件”），结合专栏源代码理解自定义组件实现方式，然后使用自定义组件的方式重构 UGC 社区首页代码。

2.1 定义页面子部件及其排列顺序

UGC 社区首页可以拆解为 2 个子部件：

- 子部件 1 ：顶部菜单栏
- 子部件 2 ：笔记列表栏

首先在 WXML 页面模板中定义 2 个子部件的容器，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 ：顶部菜单栏的切换菜单 navbar的选项卡 -->
 <view class="weui-navbar bg-white">
 </view>
 <!-- 子部件 1 ：顶部菜单栏的发表笔记按钮 navbar选项卡右侧的发表笔记图标 -->
 <view class="bg-white text-right padding-right-lg padding-tb-xs text-xl">
 </view>
 <!-- 子部件 2 ：笔记列表栏 -->
 <view>
 </view>
</view>
```

2.2 实现子部件 1 ：顶部菜单栏

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：切换菜单

多条内容交互界面，事件为用户点击屏幕事件，数据为菜单名称与当前选中菜单标志。

三个切换菜单水平排列，三个菜单分别为：

- 发现：显示所有笔记列表
- 收藏：显示用户点赞过的笔记列表
- 我的：显示用户自己发表的笔记列表

用户点击屏幕事件的事件响应为：将用户点击选中的分类名称变为绿色，其余分类名称变为灰色，同时在子部件 2 中显示用户选中的菜单对应的笔记列表。

第一次打开 UGC 社区首页时默认选中 “发现” 菜单，显示所有笔记列表，笔记列表信息从数据库获取。

显示元素 2：发表笔记按钮

静态界面，事件为用户点击屏幕事件，无数据。

在切换菜单的最右侧显示发表笔记按钮，发表笔记按钮为一个拍照图标。

用户点击屏幕事件的事件响应为：页面跳转到发表笔记页面。

2.2.1 实现显示元素 1：切换菜单

显示元素 1：切换菜单的 Navbar 实现代码与第六章第四节 “2.2 实现子部件 1 ：顶部的积分记录类型菜单” 中 “2.2.1 实现 Navbar ” 基本一致，完整实现代码请回顾第六章第四节内容或阅读专栏源代码（源代码位置见本节末尾图 10）。

菜单数据为：

```
data: {
  tabs: ["发现", "收藏", "我的"], //定义navbar的选项卡
},
```

用户点击菜单的逻辑也与第六章第四节 “2.2 实现子部件 1 ：顶部的积分记录类型菜单” 类似：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笔记列表的第一页分页数据。

事件响应函数我们定义为 `navBarTabClick` ：

```
/**
 * navbar Tab点击事件
 */
navBarTabClick: function(e) {
  //当用户单击的菜单不是当前选中菜单时,才重新设置选中菜单并重新获取笔记列表
  if (e.currentTarget.id != this.data.activeIndex) {
    this.setData({
      //重新设置用户选中的菜单为用户点击菜单
      sliderOffset: e.currentTarget.offsetLeft,
      activeIndex: e.currentTarget.id, //记录navbar的当前选中选项
      //重置数据全部加载完毕的标志
      isNoMoreData: false,
    });
    //获取用户新选择菜单的笔记列表的第一页分页数据
    this.getNoteList(true)
  }
},
```

getNoteList 方法在 2.3 中具体实现。

2.2.2 实现显示元素 2：发表笔记按钮

发表笔记按钮的用户点击事件使用 Navigator 实现：

```
<!-- 子部件 1 ：顶部菜单栏的发表笔记按钮 navbar选项卡右侧的发表笔记图标 -->
<view class="bg-white text-right padding-right-lg padding-tb-xs text-xl">
  <navigator url="../postnote/postnote">
    <!-- 发表按钮使用 ColorUI 中的照相机图标 -->
    <text class="icon-camerafill"></text>
  </navigator>
</view>
```

2.3 实现子部件 2 ：笔记列表栏

笔记列表栏包含一个显示元素：

笔记列表

多条内容交互界面，事件为用户下滑屏幕到页面底部事件与用户点击屏幕事件，数据为：属于子部件 1 选中菜单的笔记信息数组。

第一次打开 UGC 社区首页时默认选中 “发现” 分类，显示所有笔记列表。

用户下滑屏幕到页面底部事件的事件响应为：从笔记信息表中分页获取当前选中菜单的下一页笔记信息数据，并追加到笔记信息数组末尾。如果笔记信息表中当前选中菜单的所有数据都获取完毕，用户下滑屏幕将不再获取分页数据。

笔记信息数组中的笔记数据以左右两列卡片式瀑布流的方式显示，一个卡片显示一个笔记信息，每个卡片显示笔记第一张图片的缩略图、笔记文字内容的开头部分、笔记发表时间、笔记获得的点赞数。

用户点击屏幕事件的事件响应为：在用户点击笔记卡片后，页面跳转到笔记详情页面，需要向笔记详情页面传递笔记 ID。

笔记信息表的数据需要从云数据库中获取。

笔记列表的实现主要包含两部分，一是根据用户选择的菜单获取对应的笔记列表数据，二是将获取到的数据以瀑布流的方式展示在页面中。

2.3.1 获取笔记列表数据

在本节前文中我们已经知道了不同菜单的笔记列表数据不同，每个菜单需要调用不同的数据服务来获取笔记列表，具体逻辑如下：

发现菜单：显示所有笔记列表，需要调用 `NoteService` 类的 `getNoteList` 方法获取数据，传入的 openId 参数为空。

收藏菜单：显示用户点赞过的笔记列表，需要 3 个步骤才能获取数据：

- 第一步，调用 `UpvoteService` 类的 `getMyUpvoteList` 方法分页获取用户点赞记录列表；
- 第二步，将点赞记录列表中每个笔记 ID 都存放到一个全量笔记 ID 数组中，存放时去重；
- 第三步，调用 `NoteService` 类的 `getNotesByIndex` 方法，输入参数为全量笔记 ID 数组，返回结果即用户点赞过的笔记列表。

我的菜单：显示用户自己发表的笔记列表，需要调用 `NoteService` 类的 `getNoteList` 方法获取数据，传入的 openId 参数为当前用户的 OpenID（在第五章第五节“1. 用户首次打开小程序自动注册”中我们已经实现了存储当前用户的 OpenID 到全局变量中）。

```
const app = getApp() //定义app用于获取全局变量中的用户openid
```

实现分页加载数据的具体代码如下：

```
/**
 * 获取用户选中的菜单的笔记列表分页数据
 */
getNoteList(isTabChanged) {
  if (this.data.activeIndex == 0) {
    //发现菜单：显示所有笔记列表
    this.getAllNoteList(isTabChanged)
  } else if (this.data.activeIndex == 1) {
    //收藏菜单：显示用户点赞过的笔记列表
    this.getUpvotedNoteList(isTabChanged)
  } else {
    //我的菜单：显示用户自己发表的笔记列表
    this.getMyNoteList(isTabChanged)
  }
},

/**
 * 分页获取所有笔记列表
 */
getAllNoteList(isTabChanged) {
```

```
  var that = this
  //调用 getNoteList 方法获取数据，传入的 openId 参数为空字符串
  noteService.getNoteList(
    '',
    isTabChanged,
    //回调函数
    function(noteArray) {
      //填充数据到瀑布流中显示
      that.fillData(isTabChanged, noteArray)
      if (noteArray.length <= 0) {
        this.setData({
          isNoMoreData: true //设置数据全部加载完毕的标志
        })
      }
    })
},

/**
 * 分页获取户自己发表的笔记列表
 */
getMyNoteList(isTabChanged) {
  var that = this
  //调用 getNoteList 方法获取数据，传入的 openId 参数为当前用户的 OpenID
  noteService.getNoteList(
    app.globalData.openid,
    isTabChanged,
    //回调函数
    function(noteArray) {
      //填充数据到瀑布流中显示
      that.fillData(isTabChanged, noteArray)
      if (noteArray.length <= 0) {
        this.setData({
          isNoMoreData: true //设置数据全部加载完毕的标志
        })
      }
    })
},

/**
 * 分页获取用户点赞过的笔记列表
 */
getUpvotedNoteList(isTabChanged) {
  var that = this
  //第一步，调用getMyUpvoteList方法分页获取用户点赞记录列表
  upvoteService.getMyUpvoteList(
    isTabChanged,
    //回调函数
    function(upvoteArray) {
      //第二步，将点赞记录列表中每个笔记 ID 都存放到一个全量笔记 ID 数组中，存放时去重
      var indexArray = []
      for (var i in upvoteArray) {
        indexArray.push(upvoteArray[i].noteId)
      }
      //第三步，调用 getNotesByIndex方法，获取用户点赞过的笔记列表
      noteService.getNotesByIndex(
        indexArray,
        //回调函数
        function(noteArray) {
          //填充数据到瀑布流中显示
          that.fillData(isTabChanged, noteArray)
          if (noteArray.length <= 0) {
            this.setData({
              isNoMoreData: true //设置数据全部加载完毕的标志
            })
          }
        })
    })
},
```

`fillData` 方法负责填充新获取到的分页数据到瀑布流中显示，实现代码请回顾第八章第三节的 **“3.4.2 瀑布流显示”**。

除用户点击菜单事件外，以下两个事件也会获取笔记列表数据：

页面加载完毕时系统自动执行的事件

第一次打开 UGC 社区首页时默认选中“发现”菜单，显示所有笔记列表：

```
/**
 * 生命周期函数--监听页面初次渲染完成
 */
onReady: function() {
  //页面加载时获取“发现”菜单的第一页数据
  this.getNoteList(true)
},
```

用户下滑屏幕到页面底部事件

下滑到屏幕底部时，加载下一页数据：

```
/**
* 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.getNoteList(false)
  }
},
```

2.3.2 瀑布流显示

UGC 社区首页的瀑布流展示使用与第八章第三节商城首页的“3.4.2 瀑布流显示”相同的实现方式。

瀑布左右排列的 JS 逻辑 `fillData` **，显示瀑布流左侧数据** `leftList` **与右侧数据** `rightList` **的 WXML 页面模板请回顾第八章第三节的“3.4.2 瀑布流显示”。**（在本节“1. 数据服务”中，数据服务返回的分页笔记列表 `noteArray` 的数据结构，与第八章第三节数据服务返回的分页商品列表的数据结构一致，因此可复用第八章第三节已实现的瀑布流显示逻辑与 WXML 页面模板）

笔记列表显示的内容与商品列表不同，内容卡片模板需要重新设计。

在笔记列表的内容卡片模板中，显示笔记第一张图片的缩略图、笔记文字内容的开头部分、笔记发表时间、笔记获得的点赞数（笔记列表的数据结构见本节“1. 数据服务”的代码注释），并添加 Navigator 实现点击卡片跳转到笔记详情页面：

```
<!--笔记列表的内容卡片模板-->
<template name='noteCard'>
 <view class="card">
  <navigator url="../../community/note/note?index={{data.index}}">
   <image class='card-img' mode='aspectFill' style='width:{{data.itemWidth}}px;height:{{data.itemHeight}}px;' src='{{data.img}}' lazy-load>
   </image>
   <view class='text'>
    <view class='desc'>
     <rich-text nodes="{{data.content}}"></rich-text>
    </view>
    <view class="weui-flex date">
     <view class="weui-flex__item">
      {{data.date}}
     </view>
     <view>
      <text class="icon-like margin-right-xs"></text>
      <text>{{data.upvoteNum}}</text>
     </view>
    </view>
   </view>
  </navigator>
 </view>
</template>
```

实现图 9 中卡片显示效果的自定义样式见专栏源代码 `\membership-miniprogram\component\WaterFallView\WaterFallView.wxss` 中注释为 “/* 卡片样式 */” 的部分。

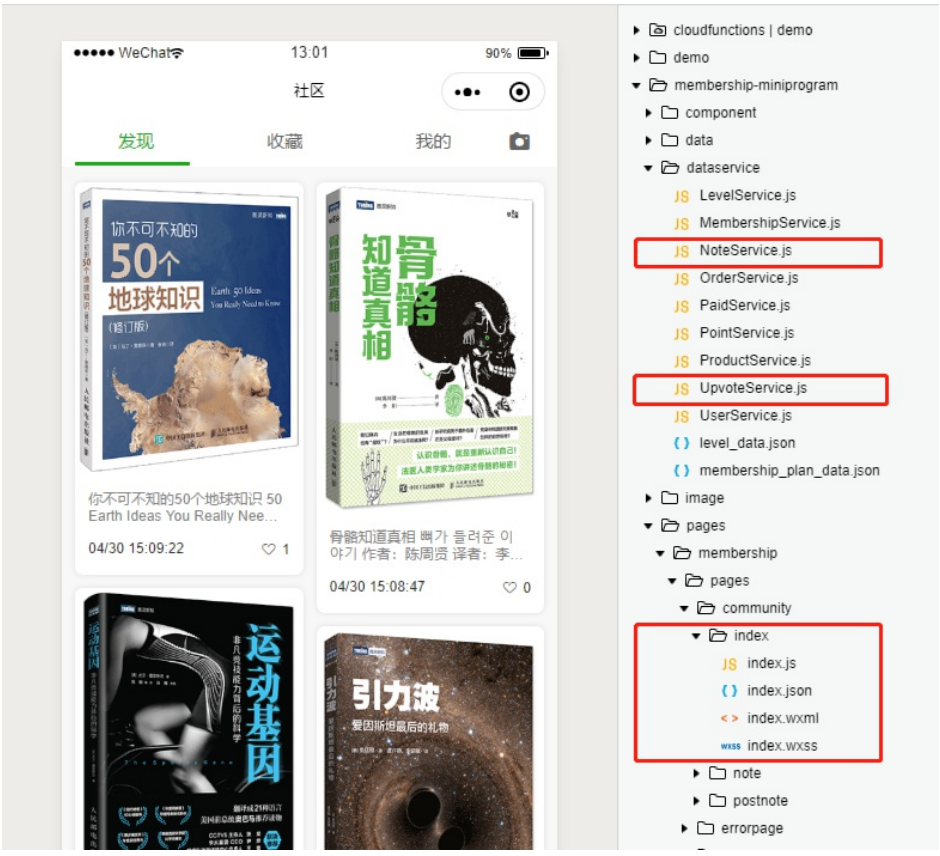
3. 专栏源代码

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 10 红框标出的位置。

图 10 本节源代码位置



下节预告

下一节，我们将开发笔记详情页面，并完成点赞功能。

实践环节

实践是通往大神之路的唯一捷径。

本节实操内容：

- 编写代码完成图 **9** 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。
- 学有余力的同学在按照本节讲解的方式实现 **UGC** 社区首页后，可以继续学习微信官方“小程序开发文档”中自定义组件的内容（文档位置：“指南” -> “自定义组件”），结合专栏源代码理解自定义组件实现方式，然后使用自定义组件的方式重构 **UGC** 社区首页代码。

}

39 发个朋友圈：开发实现发表笔记页面

点个赞：开发实现笔记详情页面与点赞功能