41 点个赞：开发实现笔记详情页面与点赞功能

更新时间：2019-09-25 17:54:23

> 构成我们学习最大障碍的是已知的东西，而不是未知的东西。
>
> —— 贝尔纳

前几节已经实现了发表笔记和笔记列表，本节将实现笔记详细内容显示及点赞功能，完成笔记详情页面开发。

在第二节中，我们已经使用 “分类拆解法” 的拆解步骤，对笔记详情页面（如图 11 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 11 笔记详情页面

1. 数据服务

在笔记详情页面加载时，需要从笔记信息表根据笔记 ID 获取笔记信息。此外，还需要从点赞记录表中获取用户是否点赞这篇笔记的信息，以决定点赞图标的显示状态。

根据笔记 ID 获取笔记信息的数据服务需要在 NoteService.js 的 `NoteService` 类中添加一个方法 `getNoteByIndex` 来实现。 `getNoteByIndex` 的具体实现代码请阅读专栏源代码的 `NoteService.js` 文件。

从点赞记录表中获取用户是否点赞这篇笔记，需要在 UpvoteService.js 的 `UpvoteService` 类中添加一个方法 `getMyUpvote` 来实现：

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

/**
 * 从数据库获取用户是否点赞笔记
 * @method getMyUpvote
 * @for UpvoteService
 * @param {string} id 笔记id
 * @param {function} successCallback(isUpvoted) 处理数据查询结果的回调函数
 * isUpvoted: 用户是否点赞过该笔记
 */
getMyUpvote(id, successCallback) {
  //在点赞记录表中查询用户点赞该笔记的记录数
  db.collection('upvote').where({
      noteId: id,
      _openid: app.globalData.openid,
    })
    .count()
    .then(res => {
      //如果在点赞记录表中记录数大于 0 则表示用户已点赞该笔记，否则用户未点赞该笔记
      var isUpvoted = res.total > 0
      //回调函数处理数据查询结果
      typeof successCallback == "function" && successCallback(isUpvoted)
    })
    .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)
    })
}
```

在用户点击点赞按钮后，需要更新笔记信息表和点赞记录表：

- 如果用户未点赞，则在点赞记录表中增加用户点赞该笔记的记录，在笔记信息表中的点赞数加一；
- 如果用户已点赞，则在点赞记录表中删除用户点赞该笔记的记录，在笔记信息表中的点赞数减一。

以上逻辑需要在 UpvoteService.js 的 `UpvoteService` 类中添加一个方法 `updateUpvote` 来实现：

```
/**
 * 更新用户点赞笔记的记录
 * @method updateUpvote
 * @for UpvoteService
 * @param {string} noteId 笔记id
 * @param {string} autherOpenid 笔记作者Openid
 * @param {function} successCallback(isUpvoted) 处理数据查询结果的回调函数
 * isUpvoted: 用户是否已点赞该笔记
 */
updateUpvote(noteId, autherOpenid, successCallback) {
  //在点赞记录表中查询用户点赞该笔记的记录
  db.collection('upvote').where({
    noteId: noteId,
    _openid: app.globalData.openid,
  })
    .get()
    .then(res => {
      //如果在点赞记录表中记录数大于 0 则表示用户已点赞该笔记
      if (res.data.length > 0) {
        //如果用户已点赞，则在点赞记录表中删除用户点赞该笔记的记录
        db.collection('upvote').doc(res.data[0]._id)
          .remove()
          .then(r => {
            //在笔记信息表中的点赞数减一
            db.collection('user_note').doc(noteId).update({
              data: {
                upvoteNum: _.inc(-1)
              }
            })
              .then(r => {
                //回调函数，返回值为 false 表示页面中显示的点赞图标应为未点赞状态
                typeof successCallback == "function" && successCallback(false)
              })
          })
      } else {
        //如果用户未点赞，则在点赞记录表中增加用户点赞该笔记的记录
        db.collection('upvote')
          .add({
            data: {
              date: db.serverDate(),
              noteId: noteId,
              autherOpenid: autherOpenid
            }
          })
          .then(r => {
            //在笔记信息表中的点赞数加一
            db.collection('user_note').doc(noteId).update({
              data: {
                upvoteNum: _.inc(1)
              }
            })
              .then(r => {
                //回调函数，返回值为 true 表示页面中显示的点赞图标应为已点赞状态
                typeof successCallback == "function" && successCallback(true)
              })
          })
      }
    })
}
```

完整的数据服务代码请阅读专栏源代码的 **NoteService.js** 文件与 **UpvoteService.js** 文件，具体代码位置见本节末尾图 **12**。

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。

从数据库读取笔记信息、点赞记录需要一定时间，需要一个从数据库获取完数据的标志 `inited` ，当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
},
```

2.1 定义页面子部件及其排列顺序

笔记详情页面可以拆解为 3 个子部件：

- 子部件 1 ：笔记图片栏
- 子部件 2 ：笔记文字内容
- 子部件 3 ：操作栏
- 子部件 4 ：底部栏

首先在 WXML 页面模板中定义 4 个子部件的容器 ，笔记正文内容可以使用 WeUI 的 Article 组件样式，底部栏可以使用 WeUI 的 Footer 组件样式：

```
<!-- 当获取到笔记数据后，再显示页面-->
<view wx:if="{{inited}}" class="bg-white">
  <!-- 子部件 1 ：笔记图片栏 -->
  <view class="text-center">
  </view>
  <!-- 笔记正文内容使用 WeUI 的 Article 组件样式 -->
  <view class="weui-article">
    <view class="weui-article__section">
      <!-- 子部件 2 ：笔记文字内容 -->
      <view class="weui-article__p">
      </view>
      <!-- 子部件 3 ：操作栏 -->
      <view class="weui-article__p margin-lr">
      </view>
    </view>
  </view>
  <!-- 子部件 4 ：底部栏 使用 WeUI 的 Footer 组件样式 -->
  <view class="weui-footer padding-bottom">
  </view>
</view>
```

2.2 实现子部件 1 ：笔记图片栏

子部件 1 包含 1 个的显示元素：

幻灯片

多条内容交互界面，事件为用户滑动屏幕事件，数据为笔记信息中的图片字段。

使用图片轮播的方式显示笔记中的所有图片，图片水平居中显示，图片缩放保持宽高比。

用户滑动屏幕事件的事件响应为：根据用户滑动屏幕的方向，在幻灯片中切换显示前一张图片或后一张图片。

笔记信息在页面加载时，通过其他页面跳转到笔记详情页面时传递的笔记 ID 从云数据库的笔记信息表中获取。

幻灯片使用小程序官方组件 swiper 实现。

笔记信息在页面加载时调用数据服务获取，页面加载时还要获取用户是否已经点赞过这篇笔记的信息。

由于用户可能上传多张大小不同的图片，在获取笔记信息时还需要计算笔记中所有图片的最大高度，用于指定 swiper 组件的高度。

```
data: {
  noteInfo: {}, //笔记信息
  isUpvoted: false, //用户是否已经点赞过这篇笔记
  maxHeight: 0, //笔记中所有图片的最大高度，用于指定swiper组件的高度
},
```

页面加载时调用数据服务的代码如下：

```
/**
 * 生命周期函数--监听页面加载
 */
onLoad: function (options) {
  var that = this
  //调用数据服务获取笔记信息
  noteService.getNoteByIndex(
    options.index, //options.index为页面跳转时传递的笔记 ID
    function (noteInfo) {
      if (noteInfo.images.length > 0) {
        var windowWidth = wx.getSystemInfoSync().windowWidth //获取手机屏幕的宽度信息
        //如果笔记中包含图片，要循环比较每个图片的高度
        //得到所有图片的最大高度，用于指定swiper组件的高度
        var maxHeight = that.data.maxHeight
        for (var i in noteInfo.images) {
          //图片的最大宽度为手机屏幕宽度，如果图片宽度大于手机屏幕宽度要等比例缩放图片高度与宽度
          if (noteInfo.images[i].width > windowWidth) {
            noteInfo.images[i].height = noteInfo.images[i].height * windowWidth / noteInfo.images[i].width
            noteInfo.images[i].width = windowWidth
          }
          //获取笔记所有图片的最大高度
          if (maxHeight < noteInfo.images[i].height) {
            maxHeight = noteInfo.images[i].height
          }
        }
        //设置笔记数据
        that.setData({
          noteInfo: noteInfo,
          maxHeight: maxHeight,
        })
      }
      //调用数据服务获取用户是否已经点赞过这篇笔记的信息
      upvoteService.getMyUpvote(
        noteInfo._id,
        function (isUpvoted) {
          //设置数据
          that.setData({
            isUpvoted: isUpvoted,
            inited: true //在读取完所有数据后，设置从数据库读取数据完成标志
          })
        })
    })
},
```

请参照前面章节的内容自行添加数据服务引用。

使用 swiper 组件实现幻灯片的 WXML 页面模板代码为：

```
<!-- 子部件 1：笔记图片栏 -->
<view class="text-center">
  <!-- 当笔记中有图片时才显示幻灯片 -->
  <swiper wx:if="{{noteInfo.images.length > 0}}" style="height: {{maxHeight + 50}}px;" indicator-dots="true" indicator-color="#888888" indicator-active-color="#353535">
    <block wx:for="{{noteInfo.images}}" wx:key="item.url">
      <swiper-item class="align-center">
        <!-- 图片使用缩放模式，宽度不变，高度自动变化，保持原图宽高比不变 -->
        <image mode="widthFix" src="{{item.url}}"  style="width: {{item.width}}px;"></image>
      </swiper-item>
    </block>
  </swiper>
</view>
```

2.3 实现子部件 2：笔记文字内容

子部件 2 包含 1 个的显示元素：

文字内容

单条内容交互界面，无事件，数据为笔记信息中的文字内容字段。

文字内容用 text 标签显示即可。

```
<!-- 子部件 2：笔记文字内容 -->
<view class="weui-article__p">
 <text>{{noteInfo.content}}</text>
</view>
```

2.4 实现子部件 3：操作栏

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：发表时间

单条内容交互界面，无事件，数据为笔记信息中的发表时间字段。

在子部件 2 的下方左侧显示笔记发表时间。

显示元素 2：点赞图标

单条内容交互界面，事件为用户点击事件，数据为点赞记录表。

在子部件 2 的下方右侧显示点赞图标。

用户点击事件的事件响应为：如果用户未点赞，则在点赞记录表中增加用户点赞该笔记的记录，点赞图标修改为已点赞状态，点赞数加一；如果用户已点赞，则在点赞记录表中删除用户点赞该笔记的记录，点赞图标修改为未点赞状态，点赞数减一。

显示元素 3：点赞数

单条内容交互界面，无事件，数据为点赞记录表。

点赞数 = 点赞记录表中该笔记的点赞记录数。

子部件 3 的左右对齐排列使用 WeUI 的 Flex 组件实现。

由于点赞图标和点赞数字较小，基于用户体验角度考虑我们应设置用户点击显示元素 2 与显示元素 3 都能触发点击事件，避免用户点击位置不准无法触发事件。

点赞图标使用 ColorUI 的图标，空心图标与实心图标分别表示未点赞和已点赞这两种状态。

具体的 WXML 页面模板代码如下：

```
<!-- 子部件 3 ：操作栏 -->
<view class="weui-article__p margin-lr">
  <view class="weui-flex">
    <!-- 显示元素 1：发表时间 -->
    <view class="weui-flex__item">
      {{noteInfo.date}}
    </view>
    <!-- 由于点赞图标和数字较小，应设置用户点击显示元素 2 与显示元素 3 都能触发点击事件 -->
    <view bindtap='onUpvoteClick' class="margin-right">
      <!-- 显示元素 2：点赞图标 -->
      <!-- 使用 ColorUI 的空心与实心图标来表示未点赞和已点赞这两种状态 -->
      <text class="{{isUpvoted? 'icon-likefill':'icon-like'}} margin-right-xs"></text>
      <!-- 显示元素 3：点赞数 -->
      <text>{{noteInfo.upvoteNum}}</text>
    </view>
  </view>
</view>
```

用户点击显示元素 2 与显示元素 3 触发点赞事件，点赞的事件响应函数调用数据服务的 `updateUpvote` 执行点赞操作，然后更新数据在界面中显示最新的点赞数字和点赞图标状态。

点赞的事件响应函数代码如下：

```
/**
 * 用户点赞事件响应函数
 */
onUpvoteClick: function() {
  var that = this
  var noteInfo = that.data.noteInfo
  //调用数据服务执行点赞操作
  upvoteService.updateUpvote(
    noteInfo._id,
    noteInfo._openid,
    //点赞操作的回调函数
    function(isUpvoted) {
      if (isUpvoted) {
        //点赞操作返回值为 true ，表示用户的这次操作为点赞
        //页面中显示的点赞图标应为已点赞状态，且界面中点赞数加一
        ++noteInfo.upvoteNum
      } else {
        //点赞操作返回值为 false ，表示用户的这次操作为取消点赞
        //页面中显示的点赞图标应为未点赞状态，且界面中点赞数减一
        --noteInfo.upvoteNum
      }
      //更新页面显示数据
      that.setData({
        noteInfo: noteInfo,
        isUpvoted: isUpvoted //用户是否已经点赞过这篇笔记
      })
    })
},
```

2.5 实现子部件 4 ：底部栏

子部件 4 包含 1 个的显示元素：

社区首页

静态界面，事件为用户点击事件，无数据。

用户点击事件的事件响应为：页面跳转到 UGC 社区首页。

底部栏使用 WeUI 的 Footer 组件样式，点击页面跳转用 Navigator 实现：

```
<!-- 子部件 4 ：底部栏 使用 WeUI 的 Footer 组件样式 -->
<view class="weui-footer padding-bottom">
  <view class="weui-footer__links">
    <navigator open-type="switchTab" url="../index/index" class="weui-footer__link">社区首页</navigator>
  </view>
</view>
```

由于篇幅所限，包含完整样式的 WXML 页面模板代码请查阅专栏源代码 note.wxml 与 note.wxss ，完整的 JS 逻辑代码见 note.js ，具体代码位置见本节末尾图 15 。

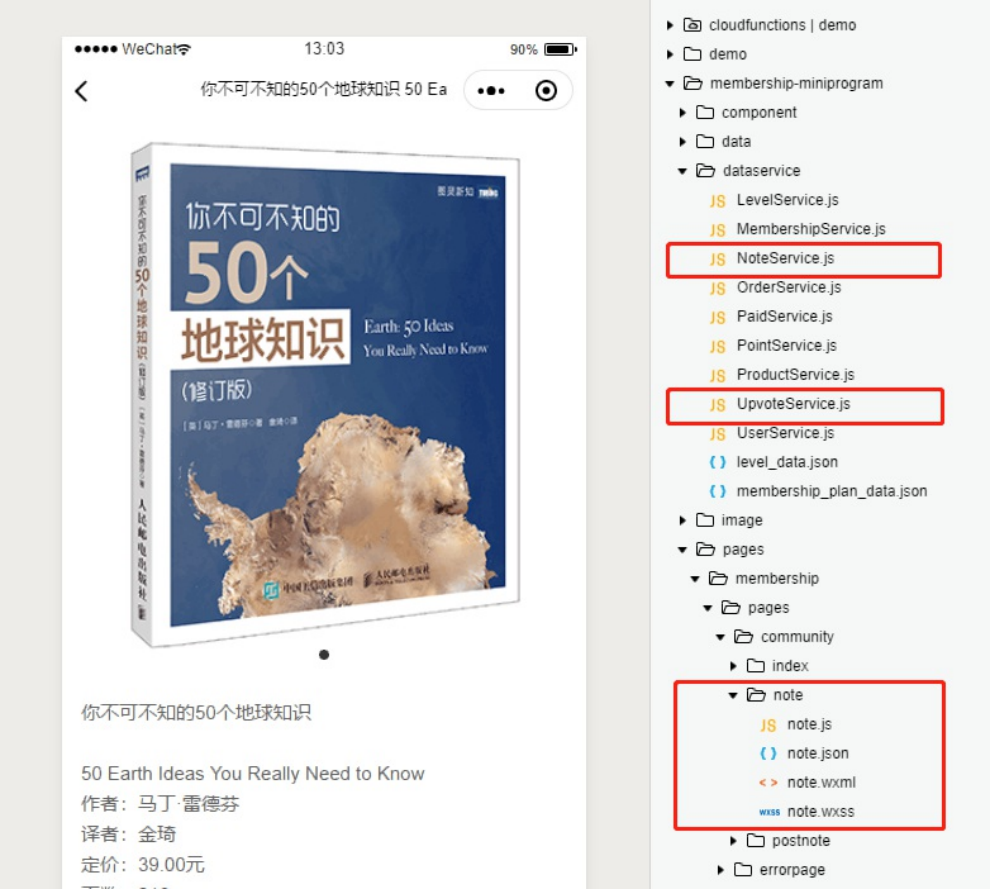
3. 专栏源代码

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 12 红框标出的位置。

图 12 本节源代码位置



下节预告

从下一节开始，我们将开始着手完成整个项目的收尾工作，包括实现个人中心页面等内容。

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 11 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。**

}

40 朋友圈刷起来：开发实现 UGC 社区首页面

42 个人中心业务设计：让用户纵览全局