38 开发实现发表笔记接口

更新时间：2019-09-18 17:47:33

> 知识犹如人体的血液一样宝贵。
>
> ——高士其

在开发 UGC 社区的页面之前，我们需要先实现发表笔记接口，然后开发发表笔记页面调用接口实现发表笔记功能，在实现发表笔记功能后 UGC 社区首页与笔记详情页面中才会有数据供显示。

本节将实现发表笔记接口，发表笔记页面在下一节实现。

在前一节中，我们已经整理了发表笔记接口的功能，包括：

- 小程序客户端：
  - 如果用户有选择图片，读取每张图片的高度与宽度，然后上传每张图片到云开发的图片存储中
  - 将用户选择的每张图片的高度、宽度、云开发中图片存储地址，用户输入的文字内容传递给云函数
- 云函数：
  - 将用户输入的文字内容和选择的图片存储到笔记记录表中
  - 在成长值获取记录表中记录用户发表笔记获得 1000 成长值
  - 用户表中的用户当前总成长值增加 1000 成长值
  - 调用成长值风控规则校验成长值记录是否有异常

根据以上功能设计，我们可以整理出发表笔记接口的程序逻辑。

1. 小程序客户端程序逻辑

由于用户发表的笔记包含图片，我们首先需要搞清楚两个问题：

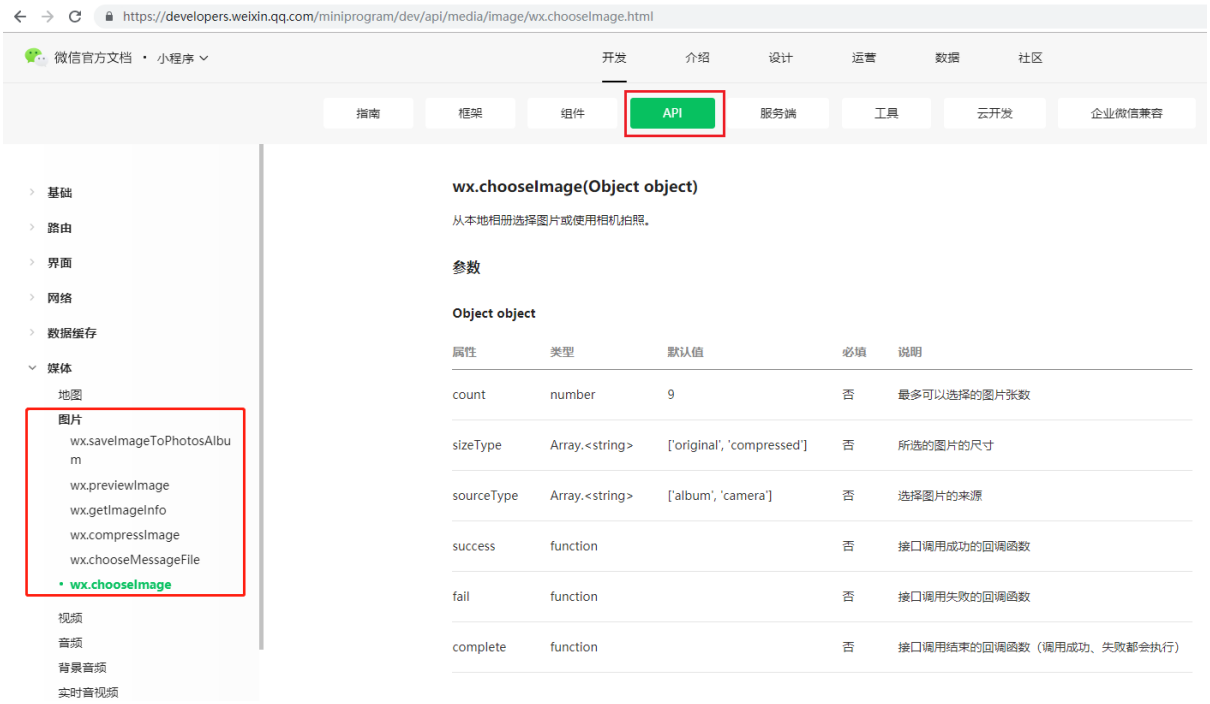
如何实现用户从相册选择图片或打开相机拍照？

通过阅读微信官方的“小程序开发文档”，我们会发现小程序官方提供了丰富的媒体 API ，使用这些 API 可以调用微信的原生能力非常方便的实现图片、音频、视频等操作。

使用小程序的图片 API 就可以实现用户从相册选择图片或打开相机拍照，微信官方的“小程序开发文档”有详细的 图片 API 说明文档，**文档位置如下：**

- 入口网址：小程序图片 API 文档，如果微信官方文档改版导致链接失效，请按图索骥。
- 入口位置：在“小程序开发文档”的“ **API** ”栏目，如图 **4** 红框标出部分。

图 **4** 小程序图片 **API** 文档位置



在微信官方的小程序示例中，对每个媒体 **API** 的具体使用都有 **Demo** 进行展示（在 **Demo** 底部“接口”菜单中），各位同学可以下载小程序示例源代码进行学习（源代码与 **Demo** 二维码请回顾第三章第一节 **“2.4** 使用小程序组件开发页面”）。

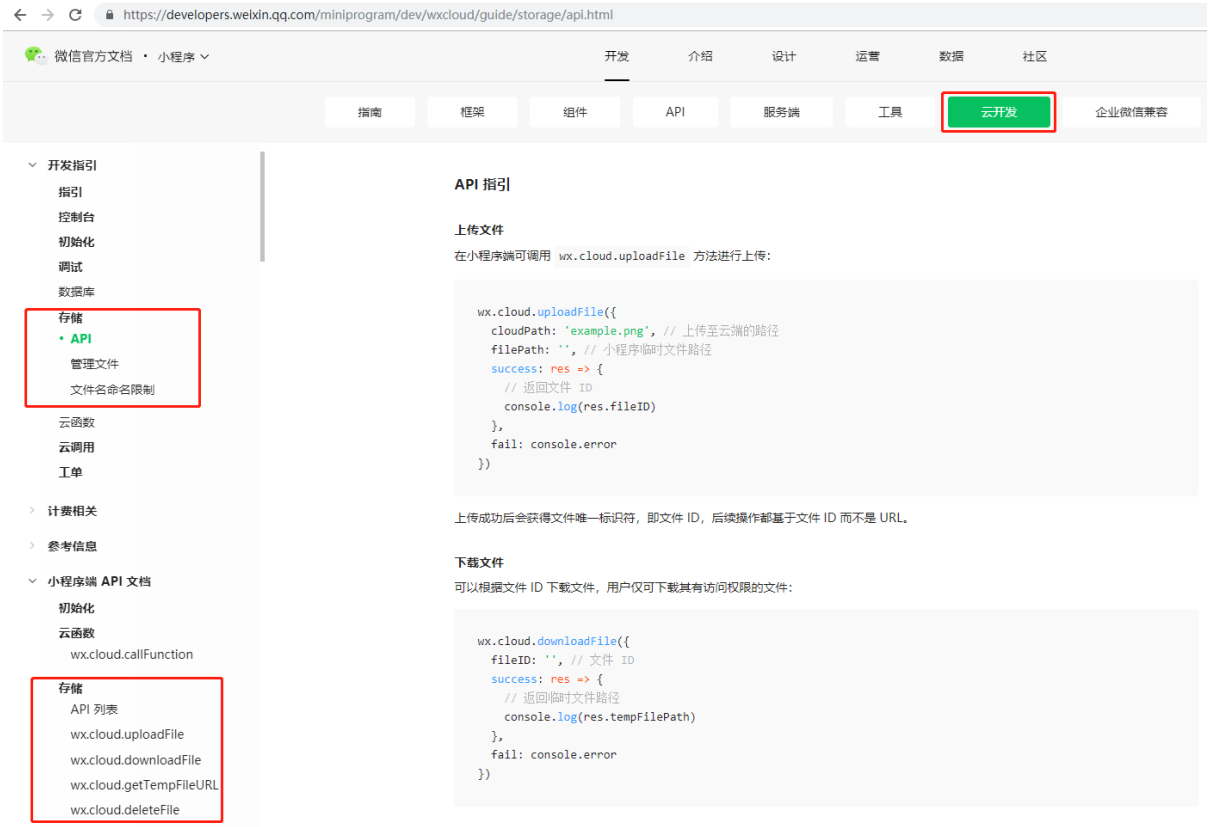
用户的图片上传到哪里进行存储？

除了云数据库和云函数之外，小程序云开发还提供了云存储的能力，**用户的图片可以上传到云开发的云存储中。**

使用云开发在云存储中保存图片很简单，微信官方的“小程序开发文档”有详细的说明文档，**小程序客户端云存储 API 的说明文档位置如下：**

- 入口网址：小程序客户端云存储 API 文档，如果微信官方文档改版导致链接失效，请按图索骥。
- 入口位置：在“小程序开发文档”的“云开发”栏目，如图 **5** 红框标出部分。

图 **5** 小程序客户端云存储文档位置



请各位同学阅读完上述 2 个文档后再继续学习本节后续内容。

在详细阅读小程序图片 API 文档与云开发的云存储文档后，我们可以整理出小程序客户端的程序逻辑：

输入参数

在发表笔记页面，我们会使用图片 API 的方法 `wx.chooseImage` 来让用户选择图片，在用户选择图片后该方法会返回用户选择的图片的本地临时文件路径列表 `tempFilePaths` ，这即是笔记的图片内容，我们可以定义输入参数 `imageList` 来接收图片内容 。

另外，用户输入的笔记文字内容也需要作为参数传递，我们可以定义输入参数 `content` 来接收文字内容 。

```
 * @param {string} content 用户输入的笔记文字内容
 * @param {array} imageList 用户选择的图片的本地临时文件路径列表
```

构造存入数据库的笔记数据结构

每条笔记包括文字内容和图片内容，图片内容是一个数组，里面每条记录为一张图片的信息，包括图片的高度、宽度和图片地址，在 UGC 首页以瀑布流的形式显示的笔记列表中需要图片的高度和宽度信息。

```
note = {
    content: , //用户输入的笔记文字内容
    images: [{
      url: , //图片地址
      width: , //图片宽度
      height: //图片高度
    },]
  }
```

上传图片到云存储

笔记中可包含图片，也可不含图片。

如果笔记中不含图片，则直接调用云函数操作数据库。

如果笔记中包含图片，需要使用图片 API 的 `wx.getImageInfo` 获得每张图片的高度 `height` 与宽度 `width` ，然后调用云存储 API 的 `wx.cloud.uploadFile` 方法将每张图片上传到云存储，并设置图片地址 `url` 为云存储地址。

调用云函数操作数据库

调用云函数在云数据库中存储笔记数据及执行其他数据库操作。

2. 云函数程序逻辑

云函数的主要程序逻辑是数据库操作，具体程序逻辑为：

输入参数

云函数的输入参数包括笔记的文字内容 `content` 和已上传到云存储的图片内容 `images` 。

```
* @param {data} 要发表的笔记内容
* {
*   data:{
*     content,   //文字内容
*     images    //已上传到云存储的图片内容
*   }
* }
```

用户的 OpenID 可以直接使用云函数的 `cloud.getWXContext()` 得到。

从用户表 `user` 中获取用户当前总成长值 `growthValue`

向笔记信息表 `user_note` 中插入一条新记录，记录用户发表的笔记信息（content、images、发表时间等）

向成长值获取记录表 `user_growth_value` 中插入一条新记录，记录用户发表笔记获得的成长值信息（发表一篇笔记获得 1000 成长值）

计算用户发表笔记后新的用户当前总成长值 `newGrowthValue` = 用户原当前总成长值 `growthValue` + 1000

在用户表 `user` 中更新用户的当前总成长值为 `newGrowthValue`

调用在第五章第五节已实现的用户成长体系风控规则云函数 `growthValueRiskControl`，校验数据库 `user_growth_value` 中该用户的成长值记录是否有异常

3. 发表笔记接口代码实现

如还未在云数据库中创建笔记信息表 `user_note` 的同学，请先在云数据中新建该表。

根据已经整理出的输入输出参数与程序逻辑，我们就可以新建发表笔记的数据库操作云函数 `postNote` ，然后在 index.js 中实现程序逻辑：

```
const cloud = require('wx-server-sdk')

// 初始化 cloud
cloud.init()
const db = cloud.database()
const _ = db.command

//发表笔记获得的成长值
const NOTEADDGROWTHVALUE = 1000

// 云函数入口函数
```

```
/**
 * 发表笔记
 * @param {data} 要发表的笔记内容
 * {
 *   data:{
 *     content,  //文字内容
 *     images    //[] 已上传到云存储的图片内容数组
 *   }
 * }
 * @return {object} 结果
 * {
 *   data,   //bool 发表笔记成功或失败
 *   errMsg  //如果发表笔记失败，该字段包含具体错误信息
 * }
 */
exports.main = async(event, context) => {
  const wxContext = cloud.getWXContext()
  //设置默认返回值
  var result = false
  var errMsg = ''
  //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]
  //向笔记信息表中插入一条新记录，记录用户发表的笔记信息
  //插入记录后获得插入的笔记 ID
  var noteId = (await db.collection('user_note')
    .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(), //发表笔记时间
        content: event.content, //笔记文字内容
        images: event.images, //笔记图片内容
        upvoteNum: 0 //点赞数，笔记发表时为0
      }
    })).id
  //向成长值获取记录表中插入一条新记录，记录用户发表笔记获得的成长值信息
  await db.collection('user_growth_value')
    .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(), //获取成长值时间
        changeGrowthValue: NOTEADDGROWTHVALUE, //获得的成长值
        operation: "发表笔记", //成长值来源
        timestamp: '',
        orderId: '',
        noteId: noteId //对应的笔记 ID
      }
    })
  //用户发表笔记后新的用户当前总成长值
  var newGrowthValue = user.growthValue + NOTEADDGROWTHVALUE
  //在用户表中更新用户的当前总成长值
  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: {
        growthValue: newGrowthValue //新的用户当前总成长值
      }
    })
  if (updateUserResult.stats.updated == 1) {
    result = true
    //调用用户成长体系风控规则云函数，校验用户的成长值记录是否有异常
    await cloud.callFunction({
      name: 'growthValueRiskControl',
      data: {
```

```
      openid: wxContext.OPENID
    }
  })
 } else {
  errMsg = "系统异常，如有疑问请联系客服"
 }

 //返回数据库操作结果
 return {
  data: result,
  errMsg: errMsg
 }
}
```

根据已经整理出的小程序客户端程序逻辑，我们可以新建一个数据服务文件 NoteService.js，在其中定义笔记信息表的数据服务 NoteService ，然后在 NoteService 中实现供发表笔记页面调用的发表笔记接口 postNote（上传图片到云存储、调用云函数操作数据库）：

```
/**
 * 添加用户笔记到数据库
 * @method postNote
 * @for NoteService
 * @param {string} content 笔记文字
 * @param {array} imageList 笔记图片地址列表（即wx.chooseImage的返回值tempFilePaths）
 * @param {function} successCallback() 处理数据查询结果的回调函数
 */
postNote(content, imageList, successCallback) {
 //构造存入数据库的图片信息数据结构，包括图片的高度、宽度和图片地址
 var images = []
 for (var i in imageList) {
  images.push({
   url: imageList[i],
   width: 0,
   height: 0
  })
 }
 //构造笔记的数据格式
 var note = {
  content: content,
  images: images
 }

 if (images.length > 0) {
  //如果笔记中包含图片，获取图片的宽度和高度并上传图片到云存储
  //使用递归调用方法来完成每张图片的处理
  var index = 0 //从第一张图片开始处理
  this._addImage(index, note, successCallback)
 } else {
  //如果不包含图片，直接调用云函数添加笔记到数据库
  this._addNote(note, successCallback)
 }
}

/**
 * 私有方法-上传图片到云开发图片存储-递归调用
 * @method _addImage
 * @for NoteService
 * @param {string} index 当前上传图片的数组序号
 * @param {object} note 笔记
 * @param {function} successCallback() 处理数据查询结果的回调函数
 */
_addImage(index, note, successCallback) {
 var that = this
 //调用图片 API 获取图片的高度宽度信息
 wx.getImageInfo({
  src: note.images[index].url,
  //图片 API 调用成功的回调函数
  success(imageInfo) {
   note.images[index].width = imageInfo.width
   note.images[index].height = imageInfo.height
```

```
      //调用云存储 API 上传图片到云存储
      wx.cloud.uploadFile({
        cloudPath: 'user/note/images/' + app.globalData.openid + '/' + new Date().getTime().toString() + '.jpg', //云存储路径
        filePath: note.images[index].url, // 图片本地路径
      })
      //图片上传成功后的操作
      .then(res => {
        note.images[index].url = res.fileID //修改图片URL为云存储地址
        //判断当前处理的图片是否为最后一张图片
        if (index < note.images.length - 1) {
          //如果还有图片，继续递归调用，上传下一张图片到云存储
          that._addImage(index + 1, note, successCallback)
        } else {
          //如果图片已全部上传到云存储，调用云函数添加笔记到数据库
          that._addNote(note, successCallback)
        }
      })
      //云存储 API 出错处理
      .catch(err => {
        //跳转出错页面
        wx.redirectTo({
          url: '../../errorpage/errorpage'
        })
        console.error(err)
      })
    },
    //图片 API 出错处理
    fail(err) {
      //跳转出错页面
      wx.redirectTo({
        url: '../../errorpage/errorpage'
      })
      console.error(err)
    }
  })
}

/**
 * 私有方法-发表笔记到云开发数据库
 * @method _addNote
 * @for NoteService
 * @param {object} note 笔记
 * @param {function} successCallback() 处理数据查询结果的回调函数
 */
_addNote(note, successCallback) {
  //调用云函数执行发表笔记操作
  wx.cloud.callFunction({
    name: 'postNote',
    data: {
      content: note.content,
      images: note.images
    },
    success: function(res) {
      if (res.result.data == true) {
        //回调函数处理数据查询结果
        typeof successCallback == "function" && successCallback()
      } else {
        //云函数返回结果为添加笔记失败，跳转出错页面
        wx.redirectTo({
          url: '../../errorpage/errorpage'
        })
        console.error(err)
      }
    },
    fail: function(err) {
      //云函数执行出错，跳转出错页面
      wx.redirectTo({
        url: '../../errorpage/errorpage'
      })
      console.error(err)
    }
  })
}
```

请注意公有方法与私有方法在 JS 中的命名区别。

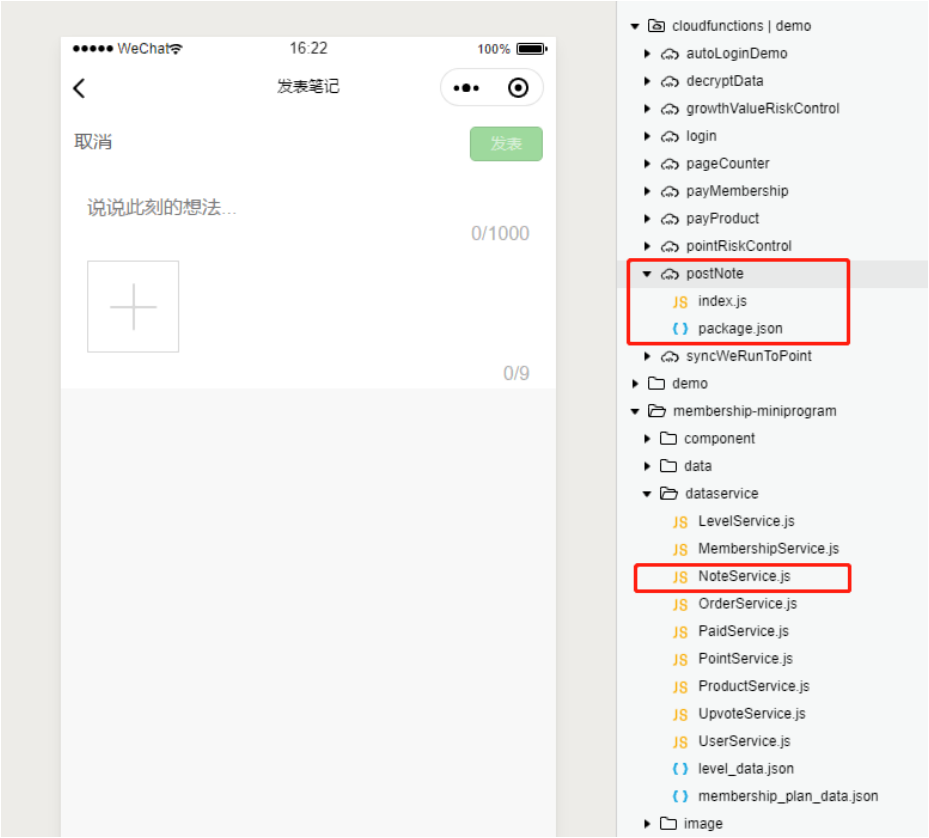
4. 专栏源代码

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 6 红框标出的位置。

图 6 本节源代码位置



下节预告

下一节，我们将调用本节编写的发表笔记接口实现发表笔记页面。

实践环节

实践是通往大神之路的唯一捷径。

本节实操内容：

- 编写代码完成发表笔记接口，如碰到问题，请阅读本专栏源代码学习如何实现。

}