第八章 二手图书网站的小程序实现

本章主要是将一个论文选题微信小程序改造成二手图书小程序。将论文选题小程序加载开发环境，分析页面之间关系以及具体功能，修改菜单名称即可完成小程序的改造。在改造过程中了解小程序的基本结构，云环境数据库的调用等。

第一节 小程序云数据库

电子商务网站需要数据库做支持，微信小程序提供付费云数据库空间，具体创建云数据库的步骤包括：(1)打开小程序开发工具，点击云开发按钮，打开云开发控制平台；(2)点击右上的设置按钮，选择环境设置菜单；(3)可以选择“管理我的环境”和“创建新环境”，就可以发现系统自动为用户创建了一个环境，点击创建新环境，用户可以创建一个环境，需要用户自己给环境命名。目前，一个微信账号只能创建两个免费的云环境，用户可以对这两个环境删除或者改名，不过腾讯公司很快可能不再提供免费运环境了。本实验创建名称为 test 的免费云环境，则如图 8.1 所示。(4)创建环境完成以后，会发现这个名称后面还有一个辅助名称，系统自动添加。(5) 点击数据库按钮，可以实现对数据权限的设置，也就是对用户读写数据库权限的设置，如图 8.2 所示。注意可以在该图显示的页面标题上看到对应的环境名称。一般来说，如果用户需要读写数据库，则开发者需要在“自定义安全规则”中配置用户具有读写权限。(6) 可以在云开发控制平台上进行存储管理，云函数管理以及运营分析等等。

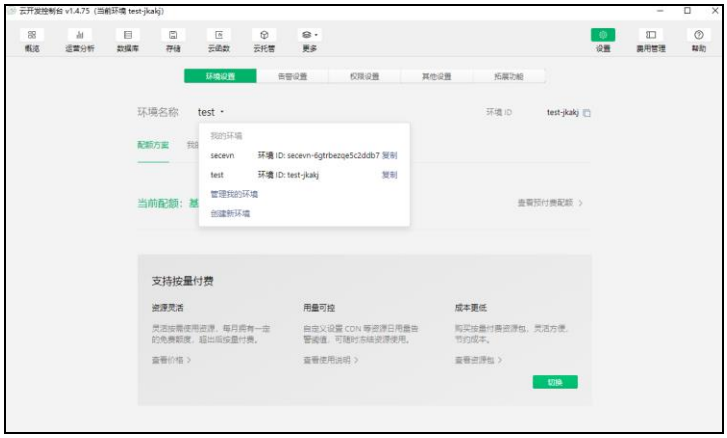

图 8.1 小程序云数据库的创建

使用数据库的时候，需要在 app.js 中配置，例如本例中云环境 ID 为“test-jkakj”，则配置示例如列表 8.1 所示，也就是配置 app.js 的代码 env 变量值，即填入运环境 ID。这个云环境 ID 决定小程序在发起云环境调用的时候（wx.cloud.xxx）会默认请求到该云环境 ID 对应的资源。可打开云控制台查看有哪些云环境 ID，如不填则使用默认环境 ID，也就是系统自动创建的环境 ID。

列表 8.1 云环境 ID 的配置

```
App({
    onLaunch: function () {
        wx.clearStorage()  //清除已经存在的会话数据
        if (!wx.cloud) {
            console.error('请使用 2.2.3 或以上的基础库以使用云能力')
        } else {
            wx.cloud.init({
                env: 'test-jkakj',
                traceUser: true,
            })
        }
        this.globalData = {}
    }
```

})

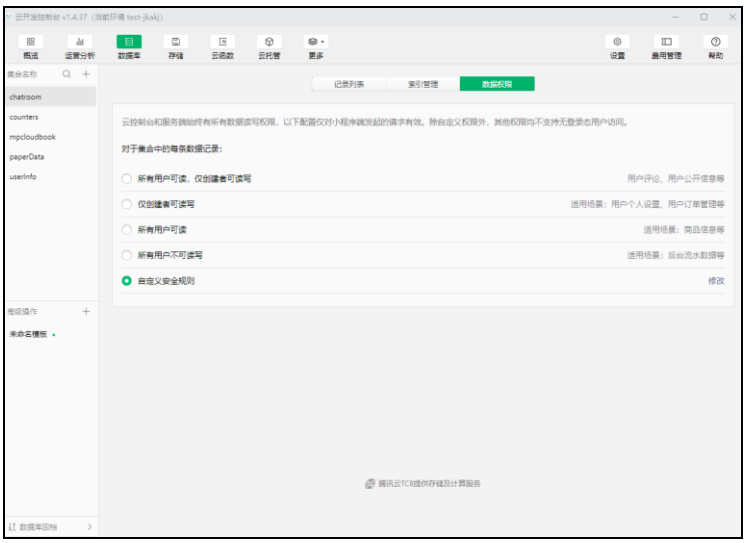

图 8.2 小程序云数据库读写权限的设置

第二节 thesisSelect 项目

本章实验开发一个小程序，小程序的名称是“论文选题”。可以通过该名称在微信中查找访问该小程序，并具体测试其各项功能，而项目代码可以通过本教材提供的链接下载。该小程序是专门开发出来供学习参考使用的，是前几章台式 PC 版二手图书网站的小程序版本。本来该小程序的主要功能应该包括用户注册，用户登录以及图书发布和图书查询等功能，但是在具体申请小程序上线的时候，不能通过审查，原因是审查人员认为这是经营性网站，必须有法人资格才行。为了避免这个问题，只能将该网站的各个功能名称做了修改。具体修改的名称为学生注册，学生登录，论文选题登记，论文题目检索等，实现功能完全一样，只是名称不同而已。这样使得小程序完全是为在校学生内部使用，因此审查通过并上线了。

在具体项目的代码中，二手图书网站的四个主要功能实现代码文件分别在对应的四个文件夹中，即 studentRegister，studentLogin，paperAdd 和 paperSelect。小程序的主页面代码文件在 index 文件夹中。在具体进行页面代码开发之前，我们需要创建数据库中的各个表，以方便后面页面实现数据库的访问。本小程序的数据库名称即对应的环境 ID 是 test-jkakj。创建了四个表，分别是 chatroom， mpcludbook，paperData 和 userInfo。这可以打开云开发控制台，点击数据库按钮，然后在弹出的对话框中创建。需要说明的是创建这四个表以后，并不需要具体设计其结构，因为它的结构在 js 文件中生成 SQL 语句中指定，列表 8.4 展示的变量名就是表中对应的字段名，我们将在下面具体讲到。

小程序的主程序配置以及各个目录在 app.json 中声明，总体样式在 app.wxss 中配置，启动运行选项在 app.js 指定，项目配置信息在 projectconfig.json 中设置。下面我们将该项目的一些主要代码文件进行简要的分析，了解这些基本的实现逻辑以后，其它小程序的功能实现都很容易理解。

一、 小程序的主体设置

app.json 中给定该小程序的所有页面路径，其中排在最前面的是主页面。主页面 index.wxml 中给出二级页面的各个链接。index.js 中的 data 实现数据的渲染，也就是该 data 数据决定页面的显示。小程序启动和关闭的时候，所有数据处理等在文件中实现。如列表 8.2 显示的 onload:function()函数实现小程序启动的时候需要完成的一些功能。

列表 8.2 小程序 app.js 文件

```
const  app  =  getApp()  //获得小程序对象
Page({
    data:  { //对应页面数据
        avatarUrl:  './user-unlogin.png',
```

```
    userInfo: {},
    logged: false,
    takeSession: false,
    requestResult: ''
  },
  onLoad: function() {    //打开页面即运行的函数
    if (!wx.cloud) {
      wx.redirectTo({
        url: '../chooseLib/chooseLib',
      })
      Return
```

二、用户注册和登录功能的实现

对照第五和六章 PC 版二手图书实现的四个主要功能，第一个功能就是用户注册，对应本项目就是学生注册，因此需要在 studentRegister 文件夹下面创建和设计和 ASP 相对应的 html 以及 asp 页面，完成相应的两个功能，即将客户端数据向服务器端提交。我们查看项目中文件夹 studentRegister 下面有四个文件，其中 studentRegister.wxml 文件对应的就是 PC 版本中用户注册 userRegister.html 文件。可以回忆综合实验五中是通过 form 标签实现 html 和 asp 页面对应的。但是对小程序而言，是不需要指明对应的文件名称的，因为这个对应文件都是在同一个目录下，也就是 studentRegister 目录下面，而且这四个文件的主文件名称相同，只是扩展文件名不同。另外，小程序规定 wxml 页面只能向 js 文件中提交数据，因此 studentRegister.wxml 提交的数据，一定是 studentRegister.js 来接收。虽然不需要在 wxml 文件中指出接收的文件名，但是需要指出接收的函数，以方便将数据提交给 js 中的哪一个函数来处理，因此查看 studentRegister.wxml 中 form 标签，就会发现 js 中接收数据的函数名称由 wxml 中 bindsubmit 来指定。用户注册页面名称为 studentRegister.wxml，文件内容如列表 8.3 所示。我们从这个代码中寻找和 html 中 form 标签的 action 属性对应的功能，应该看到就是 bindsubmit='onSubmit'，对应的就是 studentRegister.js 中的 onSubmit 函数，如列表 8.4 所示。注册功能的实现就是追加用户数据，提交后触发控件，进而运行 studentRegister.js 中的 onSubmit 函数，实现将数据向数据库表 userInfo 中的追加。同样特别注意，这里面仍然存在和 web 网站开发一样的对应问题，就是 wxml 中的 input 标签属性 name 对应的属性值，和 js 文件中 e.detail.value 后面属性的名称要一致，但是此处没有字段变量对应的问题，上面一节已经讲过了，表结构是在执行数据追加的时候自动创建，而 PC 版二手图书网站开发的时候，数据库的表结构需要事前创建出来。

列表 8.3　wxml 页面绑定 js 提交的函数名称

```
<view class='container'>
    <!--include src='../../common/head.wxml' /-->
    <view class='page-body'>
        <view class='demo-box'>
          <view class='title' >学生信息注册</view>
          <form bindsubmit='onSubmit' bindreset='onReset'>
              <input name='username' type='text' placeholder='请输入学生姓名'></input>
              <input name='password' password placeholder='请输入学生密码'></input>
              <input name='QQNum' type='text' placeholder='请输入 QQ 号码'></input>
              <input name='phoneNum' type='text' placeholder='请输入手机号码'></input>
              <button size='mini' type='default' form-type='submit'>提交</button>
              <button size='mini'  type='default' form-type='reset'>重置</button>
          </form>
        </view>
    </view>
</view>
<view class ="introduction"></view>
<include src='../common/foot.wxml' />
```

studentRegister.js 文件中，onSubmit 函数里面一定要有与 ASP 页面中的 requst.form 相同功能的函数来实现数据的获取。在小程序中，这个对应的获取数据的功能是通过资源赋值直接完成的，具体实现代码如列表 8.4 所示。从该图所示的代码中可以发现，从 wxml 传来的数据封装在 studentRegister.js 代码中的对象 e 中，我们可以直接从 e.detail.value 的属性名中获取对应的值。我们一样发现也有类似 ASP 中的数据传递对应方式。而 this.data.userName 是 studentRegister.js 页面对象的属性变量名称。

列表 8.4　小程序数据接收的实现

```
Page({
  data: {
        userName: '',
```

```
            userPassword: '',
            QQNumber: '',
            phoneNumber: '',
        },
        onSubmit: function (e) {
  //  将表中传送过来的数据赋值给 data 中的变量，通过 e 这个对象实现客户端数据的存储
          this.data.userName = e.detail.value.username;
          this.data.userPassword = e.detail.value.password;
          this.data.QQNumber = e.detail.value.QQNum;
          this.data.phoneNumber = e.detail.value.phoneNum;
          this.insertOneRecord()  //调用内部函数实现具体数据库表中的数据追加
},…
```

接收到数据以后，js 文件显然需要将接收到的数据追加到数据库中。我们可以回忆起实验五是怎样生成 SQL 语句的。由于小程序封装的很好，使得开发人员不需要生成 SQL 语句，直接将数据对应赋值即可，因此用户可以在编写代码时候模仿就可以了。如列表 8.5a 中所示，userInfo 是表的名称，我们在云环境对应的数据库中已经创建。关键词 add 含义相当于 SQL 语句中 insert，接着 data 结构中冒号，其中左边就是表结构中的字段变量名，冒号右边就是对应的 javascript 变量，里面存储从客户端获取的用户注册信息。此处字段变量名称不需要在云数据库中事先设置，字段变量在小程序运行的时候直接在数据库中创建对应的表结构。

列表 8.5a 小程序数据发送到微信云空间的实现

```
db.collection('userInfo').add({
    //data 字段表示需新增的 JSON 数据
    data: {
          userName: that.data.userName ,
          userPassword: that.data.userPassword ,
          QQNumber: that.data.QQNumber ,
          phoneNumber: that.data.phoneNumber
    },
```

总结一下就是：用户注册页面名称为 studentRegister.wxml，我们从这个代码中寻找和 html 中 form 标签的 action 属性对应的功能，应该看到就是 bindsubmit='onSubmit'，对应的就是 studentRegister.js 中的 onSubmit 函数。注册功能的实现就是追加用户数据，提交后触发控件，进而运行 studentRegister.js 中的 onSubmit 函数，实现将数据向数据库表中的追加。同样特别注意这里面仍然存在和 Web 网站一样的对应问题，就是 wxml 中的 input 标签的名称，和 js 文件中 e.detail.value 后面属性的名称要一致，但是此处没有字段变量对应的问题，因为表结构是在执行追加数据的时候自动创建的。

上面使用腾讯公司提供云数据库实现用户信息的储存，如果要保持二手图书网站后台数据的一致性，可以使用微信小程序直接访问二手图书网站的后台数据库，这时微信小程序仅仅充当一个客户端展示的功能，此时可以做如表 8.5b 所示将用户数据信息插入到后台数据库中。

列表 8.5b 小程序数据发送到网站后台的实现

```
urlData = "https://url" + "wxVersion/userRegister.asp"
wx.request({
    url: urlData,
    method: 'post',
    data:{
        userName: that.data.userName,
        userPassword: that.data.userPassword,
        QQNumber: that.data.QQNumber,
        phoneNumber: that.data.phoneNumber
    },
    header:{
        'content-type': 'application/x-www-form-urlencoded;charset=utf-8'
    },
}
```

观察列表 8.5b，可以发现 urlData 存放的就是要用户注册 asp 页面的网站地址，然后调用微信小程序的 wx.request 函数将 data 中的数据通过 http 协议 post 方法发送给用户注册 asp 页面，而该 asp 页面和 PC 版二手图书网站对应的用户注册页面几乎一致，所要修改是将返回的页面显示内容删除即可，因为此时的呈现逻辑已经交给微信小程序处理了。同时注意使用 post 方法提交数据时候，content-type 属性值是 application/x-www-form-urlencoded，字符编码是 UTF-8。如果要使用 get 方法提交数据，则需要将属性值修改为 application/json。具体详细实现，可以参阅本教程附带的二手图书微信小程序源代码。

至于用户登录功能的实现，方法上面讨论一样，只是数据库检索略有不同，下面介绍通过图书的检索来具体展示小程序如何实现数据库数据检索的。

第三节 图书的发布和检索

web 动态页面的生成是通过 asp 或者 jsp 中在 html 标签中嵌入 vb 或者 java 来具体实现的，但是小程序的逻辑主要在 js 中完成，生成的数据在 wxml 中显示出来，并不是嵌入高级语言实现。wxml 对 js 中的变量数据显示，通过{{}}两对大括号实现的。列表 8.6 是具体图书搜索结果显示示例。很显然对于一些循环语句之类，小程序作为标签给出了定义，这一点与 html 里面嵌入 javascript 等不同。用户也只要模仿使用即可，不需要记忆。

如同用户注册一样，用户使用 wxml 页面提交检索的图书名称，那么对应的 js 文件里面应该有接收的图书名称的函数，接收到数据以后，然后生成查询 SQL 语句，查询到符合条件的图书以后，再将数据返回给 wxml 页面，wxml 页面再将图书信息显示出来。列表 8.6 即是图书检索的 wxml 部分源码。可以观察到 form 标签和其内部的 input 标签，它们负责图书名称提交。

列表 8.6 图书检索的 wxml 页面

```
<view class='container'>
    <form bindsubmit='onSubmit' bindreset='onReset' class="form-con">
        <input name='paperName' type='text' placeholder='请输入论文题目，例如(电子商务)测试一下'></input>
        <button size='mini' form-type='submit' class="submit">检索</button>
        <button size='mini' form-type='reset' class="reset">重置</button>
    </form>
    <view class="table">
        <view class="tr bg-w">
            <view class="th">论文名称</view>
            <view class="th">指导教师</view>
            <view class="th">论文字数</view>
        </view>
        <scroll-view  scroll-y >
            <block wx:for="{{paperList}}" wx:key="unique">
                <view class="tr bg-g" wx:if="{{index % 2 == 0}}">
                    <view class="td">{{item.paperName}}</view>
                    <view class="td">{{item.supervisorName}}</view>
                    <view class="td">{{item.wordCount}}</view>
                </view>
                <view class="tr" wx:else>
                    <view class="td">{{item.paperName}}</view>
                    <view class="td">{{item.supervisorName}}</view>
                    <view class="td">{{item.wordCount}}</view>
                </view>
            </block>
        </scroll-view>
    </view>
</view>
```

图书检索这个文件夹下同样也有和 wxml 对应的 js 文件，接收列表 8.6 中 paperName 传来的具体图书名称。接着就要将该图书名称作为关键词到数据库中查询图书，列表 8.7 显示了具体查询图书的部分关键代码。

列表 8.7 图书检索的云数据库逻辑处理代码

```
db.collection('paperData').where({
     paperName:{
          $regex:'.*' + that.data.paperName + '.*', // '.*' 等同于 SQL 中的 '%'
          $options: 'i' }
}).get({
    success: function(res){
         that.setData({paperList: res.data})
             if(res.data.length < 1 ){
                  wx.showModal({
                       content: '没有检索到毕业论文题目', }) }
      },…
```

从列表 8.7 可以观察到将检索的结果返回给了 paperList 这个对象中，接下来就需要如何在 wxml 页面显示了，这个稍微有点麻烦。观察列表 8.6 中的标签 block，其内所含的 wx:for 就是一个循环语句，实现对 paperList 中每一条记录的显示。其中{{item.paperName}}，{{item.supervisorName}}和

{{item.wordCount}}就是显示具体三个字段对应的值。wx:key="unique"就是根据每一条记录的主键一条一条的将记录显示出来。具体的显示效果可以运行小程序查看。

当然我们也可以使用 wx.request 函数访问网站后台数据库，实现图书的发布和检索，具体实现细节和用户注册和登录完全一样。需要注意是图书的检索返回大量符合条件的图书信息数据，这些数据需要传回给微信小程序显示，因此具体处理传送数据的格式问题，微信小程序使用 JSON 格式来处理数据的，因此在图书检索对应的 ASP 页面中，需要将生成的图书数据转换成 JSON 格式的数据，具体实现代码如表 8.8 所示。生成的 JSON 数据返回给小程序后可以直接在 WXML 中显示出来。

列表 8.8 图书检索中数据 JSON 格式的生成

```
Commandstr = "select * from bookInfo where bookname like '%"&bookName&"%'"
set rs = conn.Execute(Commandstr)
data = "["
Do While Not rs.EOF
     data = data & "{"
     data = data & """" & "paperName" & """:""" & rs(0).Value & ""","
     data = data & """" & "supervisorName" & """:""" & rs(1).Value & ""","
     data = data & """" & "wordCount" & """:""" & rs(2).Value & ""","
     data = Left(data, Len(data) - 1) & "},"
     rs.MoveNext
 Loop
 data = Left(data, Len(data) - 1) & "]"
 Response.write(data)
```

使用微信小程序作为客户端访问后台网站，此时使用的域名必须备案，而且必须使用 https 协议访问网站，我们可以购买 SSL 证书安装到 ASP 虚拟主机上即可实现。

第四节 二手图书小程序的实现

找到对应的项目中的页面，通过相应修改，实现二手图书小程序开发，这需要弄清楚“论文选择”各个功能之间的关系，另外它们和 PC 版的二手图书网站是完全对应起来的，同学们只要找到对应的中文修改成二手图书网站对应的中文名称即可。最后小程序运行通过后提交到微信公众号平台，可以模拟调试运行，接着提交审核，开始小程序生命周期的管理。

下载 thesisSelect 项目源代码，使用微信小程序开发工具打开该项目以后，需要注意打开 projectconfig.json，将自己的 appID 取代该项目的 appID 以后才能打开该项目。

第五节 综合实验八

一、实验目的和要求

1 掌握微信小程序云数据库访问方法；

2 掌握微信小程序数据逻辑和动态数据渲染；

3 能够进行二手图书 PC 版网站的小程序改造。

二、实验环境与工具软件

1 微信小程序开发工具；

2 互联网环境；

3 微信运行环境。

三、实验内容和步骤

本实验的主要内容包括：(1)将 PC 版的二手图书网站中各个页面移植到小程序中来。首先要给主页面和各个子页面等确定名称，然后生成对应的小程序页面，创建数据库名称，选择使用的数据库；(2)阅读 thesisSelect 项目，理解该项目各个页面的功能以及页面四个文件之间的关联关系；(3)修改该项目的菜单以及对应的数据库表和字段的名称，调试程序通过以后，借助开发环境的上传功能，将小程序

上传到微信公众号平台，开始小程序的发布以及升级等生命周期的管理。

1.thesisSlect 项目读取

(1)打开微信小程序开发工具

(2)打开项目 thesisSelect，该项目相关资源库中下载

(3)输入同学自己的 appID

(4)阅读，分析和理解该小程序各个组成部分

2. 二手图书小程序开发

(1) 确定 PC 版二手图书和 thesisSelect 中页面对应关系

(2) 依据该小程序项目，创建二手图书小程序框架

(3) 逐个对应生成二手图书的主页面和各个分页面

(4) 拷贝和修改 thesisSelect 中的内容到二手图书项目中

(5) 编辑，调试最后运行生成二手图书小程序

(6) 上传到微信公众号平台中，对小程序进行版本管理

四、实验报告（总分 5 分）

1. 分析说明 thesisSelect 小程序的基本页面关系和功能（1 分）

2. 设计二手图书小程序的主页面和各个二级链接页面的名称以及对应功能（2 分）

3. 修改模板，生成二手图书小程序代码并运行通过(小程序运行截图)，同时将源码项目以附件形式压缩上传微信公众平台。（1 分）

4. 模拟提交审核小程序，并对其进行管理(截图说明)（1 分）