第7章：微信小程序实战

本章我们将进入微信小程序实战的部分，下面我们将开发一个图书商城的小例子来介绍微信小程序的开发流程。

7.1 项目结构介绍

首先我们使用微信开发者工具创建一个项目，并选中左上角编辑器的按钮。我们会在右边看到下图的内容：

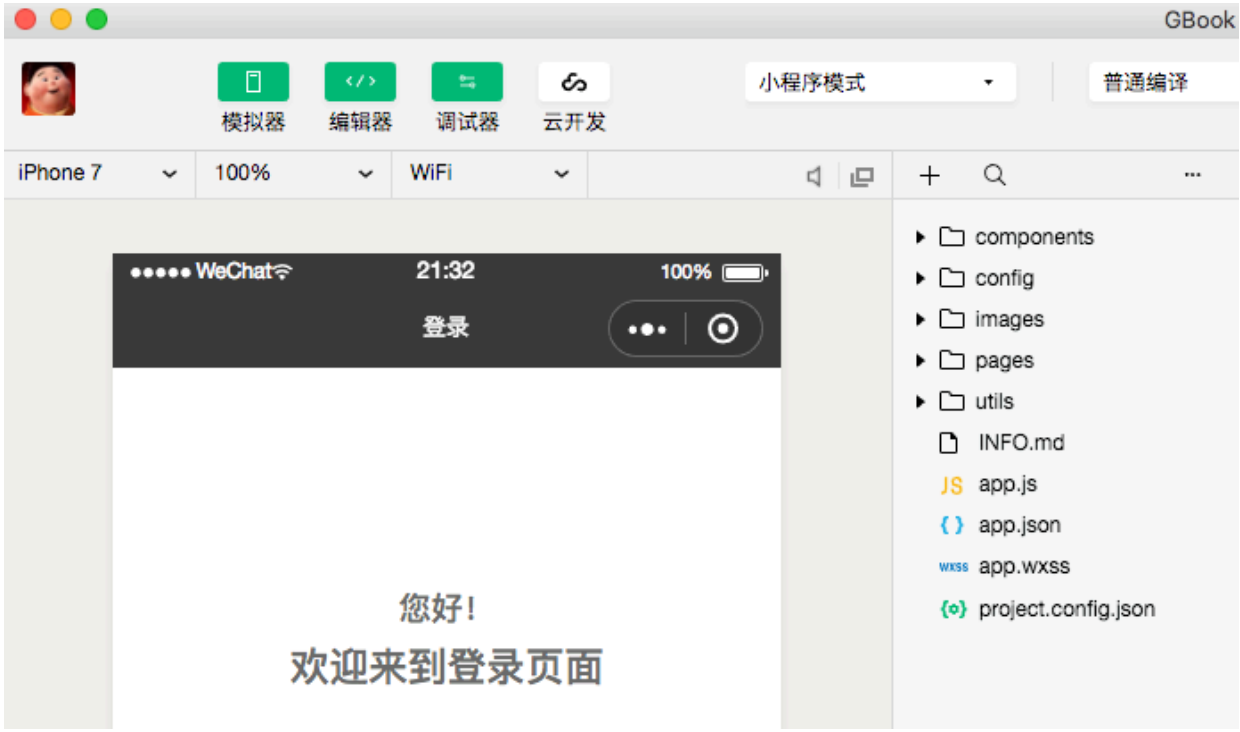

图7-1-1 微信开发者工具中的编辑器

上图内容就是我们创建好的项目的目录，其中：

- `pages` 目录中用于存放的是我们的 App 页面，比如：index，logs。
- `utils` 目录中用于存放的是工具类方法。
- `app.js` 文件是小程序的全局环境，在小程序初始化完成时回调，全局只回调一次。
- `app.json` 文件用来对小程序进行全局配置，决定页面文件的路径、窗口表现、设置网络超时时间、设置多 tab 等。
- `app.wxss` 文件用来定义小程序的全局样式，作用于每一个页面。
- `project.config.json` 文件是项目的配置文件，存放开发小程序的环境版本，appid，项目名称等。

仅仅只有上面的两个目录是不够用的，我们可以多创建几个目录来方便管理我们的代码：

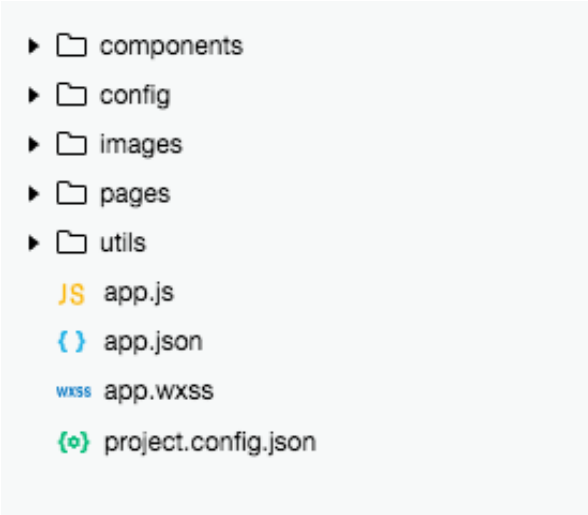

图7-1-2 项目目录

- `components` 目录用于存放我们自定义的组件（注：该项目暂未用到）。
- `config` 目录中用于存放项目的各种配置。
- `images` 目录中用于存放图片资源。

7.2 项目实战

接下来我们进入项目实战部分，我们的图书商城需求主要分为以下几个部分。

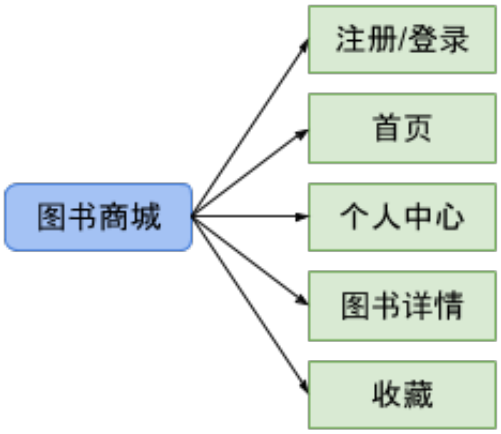

图7-2-1 项目框架

如图所示可以看到，我们的图书商城主要有登录与注册页面、首页、个人中心页面，图书详情页面、收藏页面。下面我们将一一介绍它们是如何实现的。

7.2.1 数据请求

在开始实战之前，我们先回顾一下上一章入门中网络请求的部分。示例代码如下：

```
wx.request({
    header: {//自定义响应头
        xxx: "xxx"
    },
```

```
    url: "https://www.xxx.com", //请求链接
    method: "GET",//请求方式
    data: {//请求参数
        xxx: "xxx"
    },
    success: function (res) {
        // 处理响应成功
    },
    fail: function (error) {
        // 处理响应失败
    }
});
```

所有通过网络获取的数据，都可以使用该方式请求数据。需要注意的是，服务器域名仅支持 `HTTPS`，域名不能使用 `IP` 或者 `localhost`，域名必须经过 `ICP 备案`。HTTPS 的配置和域名备案不在本书讨论的范围中，所以数据的获取使用本地数据来模拟。

7.2.2 登录与注册页面

首先我们先实现注册页面与登录页面，由于登录页面与注册页面很相似，所以这里仅展示注册页面的实现过程。先来看下两个页面效果：

图7-2-2 登录和注册页面

通过效果图可以看到，注册页面主要分为 Logo、标题、输入框和注册按钮 4 大部分组成。我们先来看下注册页面结构图：

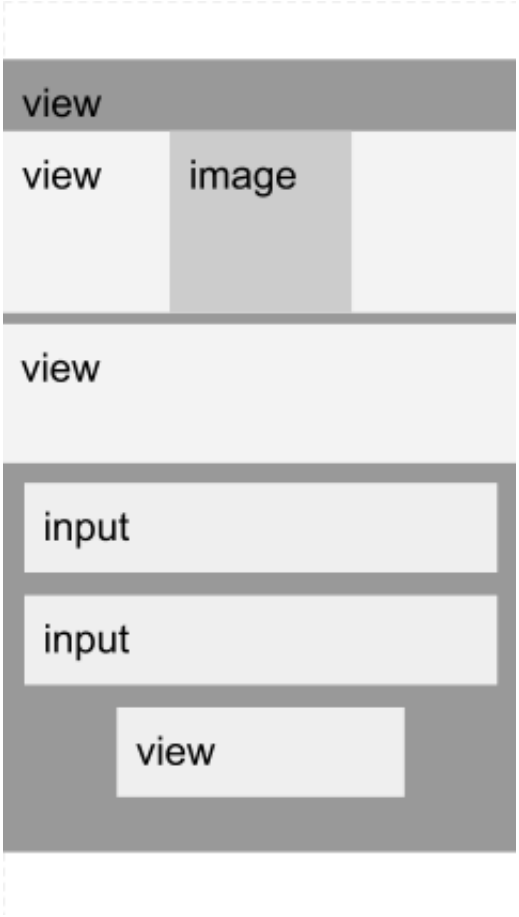

图7-2-3 注册页面结构图

从页面结构图可以看出，整个页面需要有一个大容器 `view` 水平和垂直都居中。容器中第一部分需要一个 `view` 作为 `image` 的父容器使 Logo 水平居中。容器中第二部分为标题和描述，下面是两个 `input` 输入框，一个注册按钮。我们先来看下注册页面的结构代码：

```
<!-- pages/sign-up/sign-up.wxml -->

<view>
    <view class="container">
        <view class="logo-container">
            <image class="logo"
src="/images/sign_up/bg_avatar_2x.png" aspectFill />
        </view>
        <view class="tips-container">
            <text class="title">您好! </text>
            <view class="tips">欢迎来到注册页面</view>
        </view>
        <!-- bindinput 键盘输入时触发，用于接收键盘输入的值 -->
        <input class="input-username"
               placeholder="请输入用户名"
               auto-focus
               confirm-type="next"
               bindinput="onUsernameInput" />
        <input class="input-password"
```

```
                placeholder="请输入密码"
                password
                confirm-type="done"
                bindinput="onPasswordInput" />
        <!-- bindtap 当用户点击时触发绑定的函数 -->
        <view class="btn-sign-up" bindtap="toSignUp"></view>
    </view>
</view>
```

由于微信小程序系统的控件可操作性不强，并且样式无法覆盖，所以这里使用 `view` 来实现按钮的效果。

页面中大容器水平垂直居中是注册页面中比较复杂的部分，我们来看下它是怎么实现的：

```
/* pages/sign-up/sign-up.wxss */

.container {
    /* 设置为绝对定位 */
    position: absolute;
    top: 50%;
    left: 50%;
    /* 使用 transform 将 image 向上和向左偏移 50% */
    transform: translate(-50%, -50%);
    width: 100%;
    padding: 0 80rpx;
}
```

通过 CSS 可以看到，只需要设置容器尺寸并设置为绝对定位，设置上边和左边各为自身 50% 的距离，最后使用 `transform` 将自身向上和向左各偏移 50% 这样容器就在页面中水平垂直居中了。

接下来我们来看下 Logo 是怎么居中的：

```
/* pages/sign-up/sign-up.wxss */

.logo-container {
    text-align: center;
}

.logo {
    width: 200rpx;
    height: 200rpx;
}
```

通过前面的章节学习我们知道 `image` 标签属于行内标签，所以我们只需要在其父容器中设置 `text-align: center;` 就可以了。

最后我们来看下注册页面的业务逻辑处理：

```
// pages/sign-up/sign-up.js
import config from '../../config/config.js';

Page({
    data: {
        username: "",
        password: ""
    },
    onUsernameInput: function (e) {//当用户名输入框有内容输入时被回调
        this.setData({//将输入内容保存到 username 中
            username: e.detail.value
        });
    },
    onPasswordInput: function (e) {//当密码输入框有内容输入时被回调
        this.setData({//将输入内容保存到 password 中
            password: e.detail.value
        });
    },
    toSignUp: function () {//当注册按钮点击时被调用
        if (!this.data.username) {//用户名为空，提示用户输入用户名
            wx.showToast({
                title: "请输入用户名！",
                icon: "none",
                mask: true
            });
            return;
        }
        if (!this.data.password) {//密码为空，提示用户输入密码
            wx.showToast({
                title: "请输入密码！",
                icon: "none",
                mask: true
            });
            return;
        }
        //分别将用户名，密码保存到 local storage 中
        wx.setStorageSync(config.cacheKey.username,
this.data.username);
        wx.setStorageSync(config.cacheKey.password,
this.data.password);
        wx.showToast({
            title: "注册成功请登录",
            icon: "success",
            mask: true
```

```
            });
            wx.reLaunch({//关闭所有页面，并打开登录页面
                url: '/pages/sign-in/sign-in'
            });
        }
    })
```

`config.js` 中代码如下：

```
// pages/config/config.js

module.exports = {
    cacheKey: { //配置缓存中的key，方便统一管理
        userInfo: "userInfo",
        username: "username",
        password: "password",
        favoriteBooks: "favoriteBooks",
    }
};
```

由于登录页面与注册页面比较相似，这里不再赘述。

7.2.3 首页

接下来我们来实现首页页面，先来看下效果图：

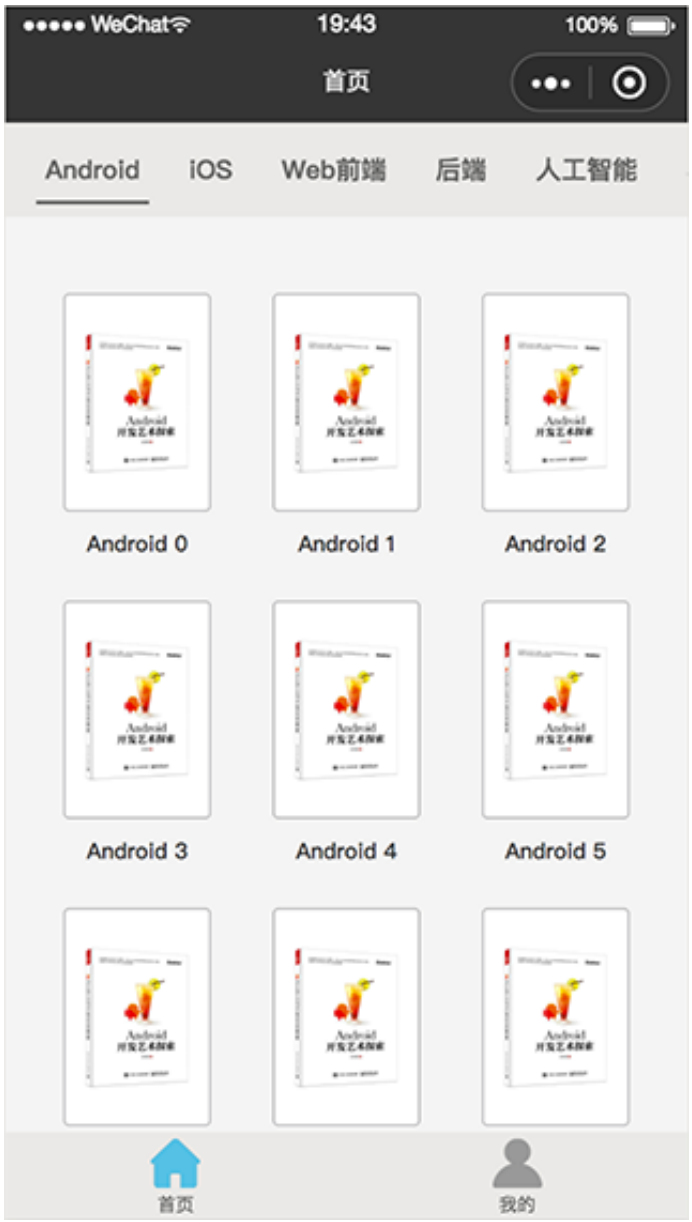

图7-2-4 首页页面

通过效果图可以看到，首页页面主要分为上下两个部分，上面部分主要是导航菜单，下面部分是一个列表。我们看下拆分后的页面结构图：

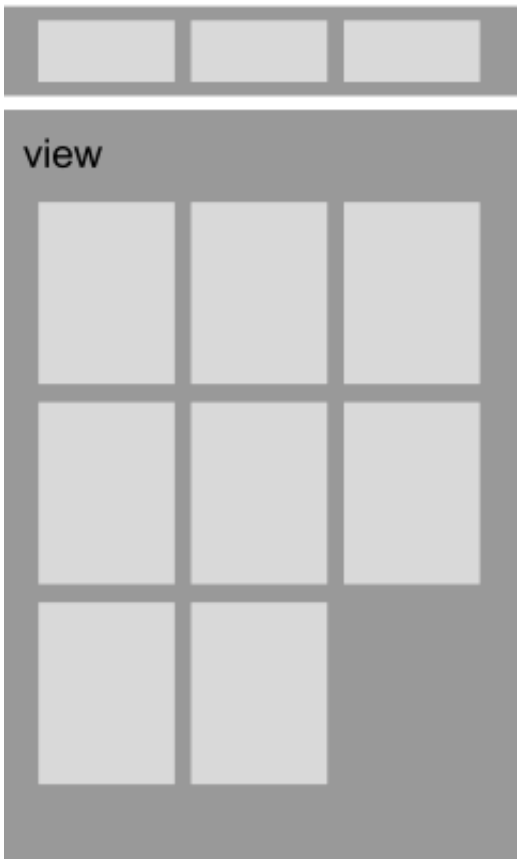

图7-2-4 首页页面结构图

从页面结构图可以看出，首页页面需要一个导航菜单和一个滑动列表。我们先来看下首页页面的结构代码：

```
<!-- pages/home/home.wxml -->

<view>
  <!-- 导航菜单 -->
  <!-- 使用 scroll-view 实现 x 轴滑动 -->
  <scroll-view scroll-x="{{true}}">
      <!-- 更具 tabList 数量动态计算导航栏的宽度 -->
      <view class="tab-menu" style="width:
{{tabList.length*150}}rpx;">
          <!-- 使用 for 循环遍历生成 item 的 view -->
          <view class="item {{currentTabIndex==index?'active':''}}"
                wx:for="{{tabList}}"
                data-item="{{item}}"
                data-index="{{index}}"
                wx:key="index"
                bindtap="onTabItemClick">
              {{item.name}}
          </view>
      </view>
  </scroll-view>
```

```
  <!-- 滑动列表 -->
  <view class="list-container">
      <!-- 使用 for 循环遍历生成 item 的 view -->
      <view class="item"
            wx:for="{{dataList[currentTabIndex]}}"
            data-item="{{item}}"
            wx:key="index"
            bindtap="onItemClick">
          <view class="cover">
              <image src="{{item.image}}" scaleToFill />
          </view>
          <view class="name">{{item.name}}</view>
      </view>
  </view>
</view>
```

由于微信小程序没有系统的导航菜单，这里需要自定义一个导航菜单，导航菜单使用 `scroll-view` 来实现，具体实现见下面代码。我们先来看下导航菜单的样式实现：

```
/* pages/home/home.wxss */

.tab-menu {
    min-width: 750rpx;
    height: 104rpx;
    background-color: #EAE9E7;
    /* 使用 flex 布局 */
    display: flex;
    /* 方向为纵向 */
    flex-direction: row;
    /* 设置为不可换行 */
    flex-wrap: nowrap;
    /* 从左边开始布局 */
    justify-content: flex-start;
    /* 垂直居中 */
    align-content: center;
    align-items: center;
    padding: 0 34rpx;
}

.tab-menu .item {
    height: 74rpx;
    line-height: 74rpx;
    padding: 0 10rpx;
    margin-right: 34rpx;
    font-size: 28rpx;
    font-family: SourceHanSansCN-Medium;
```

```
    font-weight: bold;
    color: rgba(99, 99, 98, 1);
}

.tab-menu .active {
    border-bottom: #636362 solid 4rpx;
}
```

可以看到导航菜单使用了 `Flex` 布局，这也是前面章节介绍过的知识，不清楚的同学可以翻一下前面的章节。这里需要注意的就是在页面结构代码中根据 `tableList` 动态计算宽度的部分。

我们接着来看滑动列表的实现，基于小程序的特性我们只需要使用 `Flex` 进行布局将每个 `item` 追加到页面上即可，当页面 item 数量足够多时就会把页面撑开，页面也就可以滚动了。我们来看下滑动列表的样式实现：

```
/* pages/home/home.wxss */

.list-container {
    width: 100%;
    padding: 60rpx 0 60rpx 60rpx;
    /* 使用 flex 布局 */
    display: flex;
    /* 方向为纵向 */
    flex-direction: row;
    /* 设置为可换行 */
    flex-wrap: wrap;
    /* 从左边开始布局 */
    justify-content: flex-start;
    /* 垂直居中 */
    align-content: center;
    align-items: center;
}

.list-container .item {
    /* 这里使用到了 css 中的 calc 计算函数，详见下面介绍 */
    width: calc((100% - 180rpx) / 3);
    margin-right: 60rpx;
    margin-top: 20rpx;
    border-radius: 10rpx;
    padding: 5rpx;
}

.list-container .item .cover {
    width: 100%;
    height: 240rpx;
```

```
    border: 2rpx solid #B9B9BB;
    background-color: #ffffff;
    border-radius: 6rpx;
}

.list-container .item .cover image {
    width: 100%;
    height: 100%;
}

.list-container .item .name {
    text-align: center;
    height: 70rpx;
    font-size: 24rpx;
    font-family: SourceHanSansCN-Medium;
    font-weight: 500;
    color: rgba(52, 52, 52, 1);
    line-height: 70rpx;
}
```

> 注：**calc()** = calc(四则运算)，用于动态计算长度值。
>
> - 需要注意的是，运算符前后都需要保留一个空格，例如：`width: calc(100% - 10px);`
> - 任何长度值都可以使用 calc() 函数进行计算；
> - calc() 函数支持 "+", "-", "*", "/" 运算；
> - calc() 函数使用标准的数学运算优先级规则。

最后我们来看下首页页面的业务逻辑处理：

```
// pages/home/home.js
import config from '../../config/config.js';
import data from '../../utils/data.js';

Page({
    data: {
        tabList: [],
        currentTabIndex: 0, //当前选择的 tab
        dataList: []
    },
    onLoad: function (options) {
        //读取用户登录信息
        let userInfo = wx.getStorageSync(config.cacheKey.userInfo);
        if (userInfo) { //用户已登录，则直将用户信息保存到全局变量中
            getApp().globalData.userInfo = userInfo;
            this.toLoadData();
        } else {
```

```
            wx.reLaunch({ //用户未登录，则直接跳转至登录页面
                url: "/pages/sign-in/sign-in"
            });
        }
    },
    toLoadData: function () {
        this.setData({
            tabList: data.tabList,
            dataList: data.dataList
        });
    },
    onTabItemClick: function (e) {
        console.error(e);
        let item = e.currentTarget.dataset.item;
        let index = e.currentTarget.dataset.index;
        this.setData({
            currentTabIndex: index
        });
    },
    onItemClick: function (e) {
        let item = e.currentTarget.dataset.item;
        wx.navigateTo({ //通过 url 传递参数，是不是跟 html 很像?
            url: "/pages/detail/detail?id=" + item.id + "&name=" +
item.name
        });
    }
});
```

首页页面底部有一条 `tabbar`，tabbar 需要在 `app.json` 中配置下：

```
{
    "...": "省略非主要部分",
    "tabBar": {
        "color": "#636362",
        "selectedColor": "#636362",
        "backgroundColor": "#EAE9E7",
        "borderStyle": "white",
        "list": [
            {
                "pagePath": "pages/home/home",
                "iconPath": "images/tabbar/ic_home_normal_2x.png",
                "selectedIconPath":
"images/tabbar/ic_home_selected_2x.png",
                "text": "首页"
            },
            {
```

```
                "pagePath": "pages/mine/mine",
                "iconPath": "images/tabbar/ic_mine_normal_2x.png",
                "selectedIconPath":
"images/tabbar/ic_mine_selected_2x.png",
                "text": "我的"
            }
        ]
    }
}
```

7.2.4 个人中心页面

接下来我们来实现个人中心页面，先来看下效果图：

图7-2-4 个人中心页面

通过效果图可以看到，个人中心页面主要分为两大部分，上面个人信息部分和下面滑动列表部分。通过我们来看下拆分后的页面结构图：

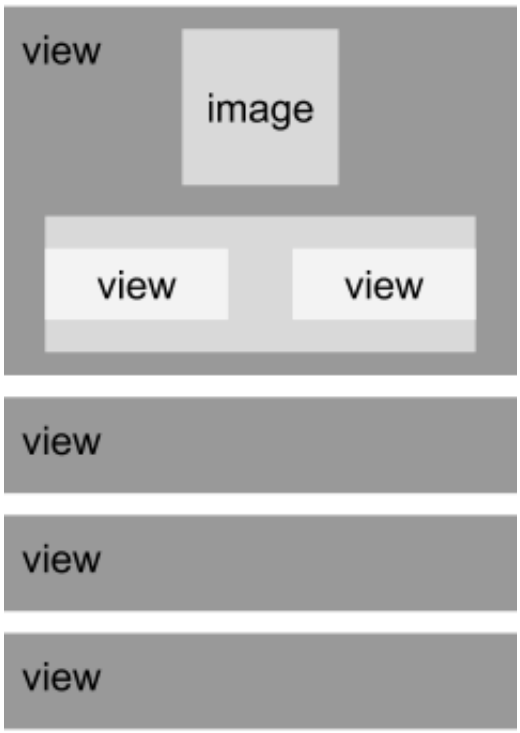

图7-2-4 个人中心页面结构图

从页面结构图我们可以看出：个人信息展示部分需要一个父容器，头像水平居中。头像下面是推荐卡部分，可以使用 `Flex` 布局来实现推荐卡。下面是功能列表比较简单。我们先来看下个人中心页面的结构代码：

```
<!-- pages/mine/mine.wxml -->

<view>
    <!-- 个人信息 -->
    <view class="info-container">
        <view class="username">{{userInfo.username}}</view>
        <view class="avatar-container">
            <image class="avatar" src="/images/logo.png" aspectFill
/>
        </view>
        <!-- 推荐卡 -->
        <view class="card-container">
            <view class="left">
                <view class="title">强力推荐卡</view>
                <view class="tips">最给力的书单都在这里</view>
            </view>
            <view class="button"></view>
        </view>
    </view>
```

```
    <!-- 滑动列表 -->
    <view class="list-container">
        <view class="item favorite">我想要的书籍</view>
        <view class="item collection" bindtap="toFavorite">我收藏的书
籍</view>
        <view class="item settings" bindtap="toSettings">设置</view>
    </view>
</view>
```

上面个人信息展示部分中头像居中的效果，我们在注册页面章节已经介绍过了现在还有印象吗？。比较复杂的是推荐卡了，我们来看下推荐卡的样式实现：

```
/* pages/mine/mine.wxss */

.info-container .card-container {
    width: 715rpx;
    height: 280rpx;
    margin: 8rpx auto;
    background-image:
url(https://raw.githubusercontent.com/jeanboydev/GBook/master/images/
mine/bg_mine_hot_2x.png);
    background-repeat: no-repeat;
    background-size: 715rpx 280rpx;
    /* 使用 flex 布局 */
    display: flex;
    /* 方向为纵向 */
    flex-direction: row;
    /* 设置为可换行 */
    flex-wrap: wrap;
    /* 水平居中 */
    justify-content: center;
    /* 垂直居中 */
    align-content: center;
    align-items: center;
}

.info-container .card-container .title {
    font-size: 33rpx;
    font-family: SourceHanSansCN-Medium;
    font-weight: bold;
    color: rgba(255, 255, 255, 1);
    line-height: 64rpx;
}

.info-container .card-container .tips {
    font-size: 28rpx;
```

```
    font-family: SourceHanSansCN-Medium;
    font-weight: 500;
    color: rgba(255, 255, 255, 1);
    line-height: 64rpx;
}

.info-container .card-container .button {
    margin-left: 120rpx;
    width: 203rpx;
    height: 63rpx;
    background-image:
url(https://raw.githubusercontent.com/jeanboydev/GBook/master/images/
mine/btn_get_2x.png);
    background-repeat: no-repeat;
    background-size: 203rpx 63rpx;
}
```

通过 CSS 可以看到，头像居中效果依然是在其父容器中设置为 `text-align: center;` 即可。推荐卡的效果也是前面介绍过的 `Flex` 布局的应用。

接下来我们来看下功能列表实现的：

```
/* pages/mine/mine.wxss */

.list-container {
    width: 100%;
}

.list-container .item {
    width: 100%;
    height: 160rpx;
    padding: 0 91rpx;
    line-height: 160rpx;
    margin-top: 16rpx;
    background-color: rgba(255, 255, 255, 1);
    font-size: 28rpx;
    font-family: SourceHanSansCN-Medium;
    font-weight: 500;
    color: rgba(99, 99, 98, 1);
}

.list-container .favorite {
    background-image:
url(https://raw.githubusercontent.com/jeanboydev/GBook/master/images/
mine/ic_favorite_2x.png);
    background-repeat: no-repeat;
    background-size: 44rpx 38rpx;
```

```
    background-position: 31rpx center;
}

.list-container .collection {
    background-image:
url(https://raw.githubusercontent.com/jeanboydev/GBook/master/images/
mine/ic_collect_2x.png);
    background-repeat: no-repeat;
    background-size: 30rpx 41rpx;
    background-position: 31rpx center;
}

.list-container .settings {
    background-image:
url(https://raw.githubusercontent.com/jeanboydev/GBook/master/images/
mine/ic_agree_2x.png);
    background-repeat: no-repeat;
    background-size: 43rpx 43rpx;
    background-position: 31rpx center;
}
```

怎么样是不是很简单？都是很常见的 CSS 样式组合。这里有一点是需要注意的在微信小程序中 `<image>` 标签和 `background-image` 都可以用来展示图片，但只有 `<image>` 的 `src` 属性可以写本地的路径，`background-image` 的路径只能写线上路径，使用本地链接编译会报错。

最后我们来看下个人中心页面的业务逻辑处理：

```
// pages/mine/mine.js

Page({
    data: {
        userInfo: null
    },
    onLoad: function (options) {
        this.setData({ //读取全局数据保存到当前页面中
            userInfo: getApp().globalData.userInfo
        });
    },
    onReady: function () { },
    onShow: function () { },
    onHide: function () { },
    onUnload: function () { },
    toFavorite: function () {
        wx.navigateTo({
            url: '/pages/favorite/favorite'
```

```
            });
        },
        toSettings: function () {
            wx.navigateTo({
                url: '/pages/settings/settings'
            });
        }
    })
```

7.2.5 图书详情页面

接下来我们来实现图书详情页面，先看下效果图：

图7-2-5 图书详情页面

通过效果图可以看到，页面主要分为图书封面、图书标题、图书介绍、图书评论、底部评论框 5 大部分。接下来我们看下页面结构图一步一步实现它：

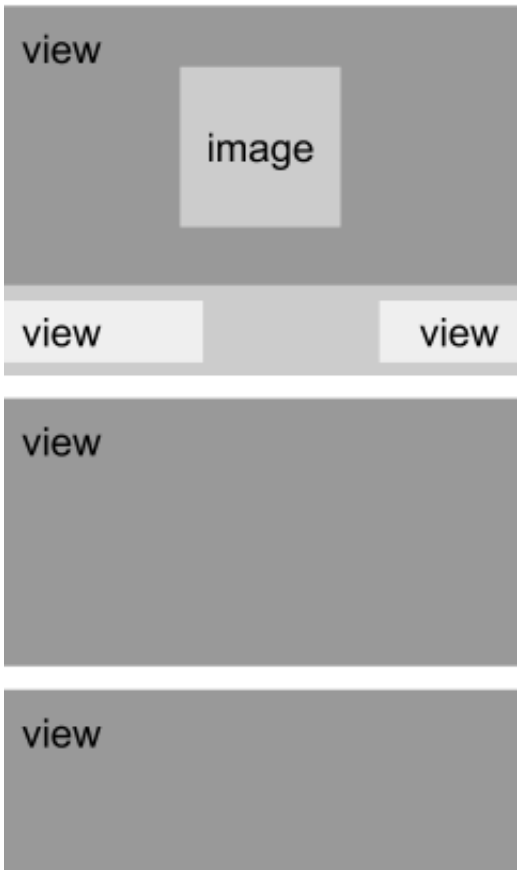

图7-2-5 图书详情页面结构图

从页面结构中可以看出：图书封面需要一个 `view` 父容器，中间 `image` 水平垂直居中显示；图书标题需要一个 `view` 父容器，标题居左、收藏居右显示；图书介绍只需要一个容器直接展示内容即可；图书评论需要一个 `view` 父容器，每条评论也需要一个父容器单独布局；底部评论框需要一个 `view` 父容器，输入框居左，按钮居右显示。

分析完了页面结构，我们先来看下图书详情页面的结构代码：

```
<!-- pages/detail/detail.wxml -->
<view>
  <!-- 图书封面 -->
  <view class="cover">
      <image src="{{bookInfo.image}}" mode="aspectFit" />
  </view>
  <!-- 图书标题 -->
  <view class="title-container">
      <view class="title">{{bookInfo.name}}</view>
      <view class="favorite {{isMarked?'selected':'normal'}}"
            bindtap="toFavorite">
          {{isMarked?'已收藏':'收藏'}}
      </view>
  </view>
  <!-- 图书介绍 -->
  <view class="introduce">{{bookInfo.introduce}}</view>
```

```
  <!-- 图书评论 -->
  <view class="comment-title-container">评论</view>
  <view class="comment-info-container">
      <block wx:if="{{commentList.length>0}}">
          <view class="item"
                wx:for="{{commentList}}"
                data-item="{{item}}"
                wx:key="index"
                bindtap="onItemClick">
              <view class="avatar"></view>
              <view class="right">
                  <view class="text">
                      <view class="username">{{item.username}}</view>
                      <view class="content">{{item.content}}</view>
                  </view>
                  <view class="agree">0</view>
              </view>
          </view>
      </block>
      <view wx:else>暂无评论</view>
  </view>
</view>
<!-- 底部评论框 -->
<view class="comment-container">
  <input class="input-comment"
         placeholder="请输入评论信息"
         confirm-type="done"
         bindinput="onCommentInput"
         value="{{currentComment}}" />
  <button class="btn-submit" bindtap="toSubmitComment">完成</button>
</view>
```

我们先来看下图书封面的是怎么实现的：

```
/* pages/detail/detail.wxss */

.cover {
    width: 100%;
    height: 414rpx;
    /* 设置为相对定位 */
    position: relative;
    background-color: #ffffff;
}

.cover image {
    width: 235rpx;
```

```
    height: 306rpx;
    /* 设置为绝对定位 */
    position: absolute;
    top: 50%;
    left: 50%;
    /* 使用 transform 将 image 向上和向左偏移 50% */
    transform: translate(-50%, -50%);
}
```

通过 CSS 可以看到我们需要先设置父容器的尺寸并设置为相对定位，然后设置图片尺寸并设置为绝对定位，设置上边和左边各为自身 50% 的距离，最后使用 `transform` 将自身向上和向左各偏移 50% 这样图片就在父容器中水平垂直居中了。这种方式是不是很熟悉，其实就是注册页面大容器居中时用的方式。

接下来我们看下图书标题的实现方式：

```
/* pages/detail/detail.wxss */

.title-container {
    /* 使用 flex 布局 */
    display: flex;
    /* 方向为纵向 */
    flex-direction: row;
    /* 设置为不可换行 */
    flex-wrap: nowrap;
    /* 从左右两端开始布局，中间留间隙 */
    justify-content: space-between;
    /* 垂直居中 */
    align-content: center;
    align-items: center;
    padding: 0 40rpx;
    height: 133rpx;
    background-color: #ffffff;
    border-top: 6rpx dashed rgba(130, 129, 129, 1);
}

.title-container .title {
    font-size: 36rpx;
    font-family: SourceHanSansCN-Medium;
    font-weight: 500;
    color: rgba(99, 99, 98, 1);
    line-height: 133rpx;
}

.title-container .favorite {
    height: 133rpx;
```

```
    font-size: 33rpx;
    font-family: SourceHanSansCN-Medium;
    font-weight: 500;
    color: rgba(99, 99, 98, 1);
    line-height: 133rpx;
    padding-right: 66rpx;
}

.title-container .normal {
    background-image:
url(https://raw.githubusercontent.com/jeanboydev/GBook/master/images/
detail/ic_favorite_normal_2x.png);
    background-repeat: no-repeat;
    background-size: 46rpx 46rpx;
    background-position: right center;
}

.title-container .selected {
    background-image:
url(https://raw.githubusercontent.com/jeanboydev/GBook/master/images/
detail/ic_favorite_selected_2x.png);
    background-repeat: no-repeat;
    background-size: 46rpx 46rpx;
    background-position: right center;
}
```

图书标题主要是使用 `Flex` 布局来实现标题居左，收藏居右的效果。

我们接着来看图书介绍和图书评论的实现方式：

```
/* pages/detail/detail.wxss */

.introduce {
    padding: 40rpx;
    font-size: 28rpx;
    font-family: SourceHanSansCN-Medium;
    font-weight: 500;
    color: rgba(99, 99, 98, 1);
    line-height: 64rpx;
}

.comment-title-container {
    padding: 0 40rpx;
    height: 133rpx;
    background-color: #ffffff;
    font-size: 33rpx;
    font-family: SourceHanSansCN-Medium;
```

```
    font-weight: 500;
    color: rgba(99, 99, 98, 1);
    line-height: 133rpx;
    border-bottom: 6rpx dashed rgba(130, 129, 129, 1);
}

.comment-info-container {
    padding: 40rpx 40rpx 200rpx 40rpx;
    background-color: #ffffff;
    font-size: 33rpx;
    font-family: SourceHanSansCN-Medium;
    font-weight: 500;
    color: rgba(99, 99, 98, 1);
}

.comment-info-container .item {
    height: 140rpx;
    /* 使用 flex 布局 */
    display: flex;
    flex-direction: row;
    flex-wrap: nowrap;
    justify-content: flex-start;
    align-content: center;
    align-items: center;
}

.comment-info-container .item .avatar {
    width: 89rpx;
    height: 89rpx;
    background-image:
url(https://raw.githubusercontent.com/jeanboydev/GBook/master/images/
logo.png);
    background-repeat: no-repeat;
    background-size: 89rpx 89rpx;
    border-radius: 89rpx;
}

.comment-info-container .item .right {
    /* 与 flex 布局结合使用，表示该子元素填充剩余的空间 */
    flex-grow: 1;
    margin-left: 10rpx;
    display: flex;
    flex-direction: row;
    flex-wrap: nowrap;
    justify-content: space-between;
    align-content: center;
    align-items: center;
```

```
}

.comment-info-container .item .right .text {
    /* 嵌套使用 flex 布局 */
    display: flex;
    flex-direction: column;
    flex-wrap: nowrap;
    justify-content: center;
    align-content: center;
    align-items: flex-start;
}

.comment-info-container .item .right .text .username {
    font-size: 28rpx;
    font-family: SourceHanSansCN-Medium;
    font-weight: 500;
    color: rgba(130, 129, 129, 1);
}

.comment-info-container .item .right .text .content {
    font-size: 33rpx;
    font-family: SourceHanSansCN-Medium;
    font-weight: 500;
    color: rgba(99, 99, 98, 1);
    margin-top: 10rpx;
}

.comment-info-container .item .right .agree {
    font-size: 28rpx;
    font-family: SourceHanSansCN-Medium;
    font-weight: 500;
    color: rgba(130, 129, 129, 1);
    line-height: 64rpx;
    padding-right: 65rpx;
    background-image:
url(https://raw.githubusercontent.com/jeanboydev/GBook/master/images/
detail/ic_agree_2x.png);
    background-repeat: no-repeat;
    background-size: 45rpx 45rpx;
    background-position: right center;
}
```

图书介绍比较简单这里不再赘述。图书评论主要也是 `Flex` 布局的应用，细心的读者可以发现灵活的使用 `Flex` 布局，可以满足绝大部分界面结构。

我们来看下底部评论框的实现：

```
/* pages/detail/detail.wxss */

.comment-container {
    width: 100%;
    height: 116rpx;
    background-color: #EAE9E7;
    /* 使用 fixed 定位使 view 固定在底部 */
    position: fixed;
    left: 0;
    bottom: 0;
    /* 使用 flex 布局 */
    display: flex;
    /* 方向为纵向 */
    flex-direction: row;
    /* 设置为不可换行 */
    flex-wrap: nowrap;
    /* 从左右两端开始布局，中间留间隙 */
    justify-content: space-between;
    /* 垂直居中 */
    align-content: center;
    align-items: center;
}

.comment-container .input-comment {
    flex-grow: 1;
    height: 70rpx;
    border: 1rpx dashed #D3D0D0;
    margin-left: 20rpx;
    font-size: 28rpx;
    font-family: SourceHanSansCN-Medium;
    font-weight: 500;
    color: rgba(130, 129, 129, 1);
    line-height: 70rpx;
    padding: 0 20rpx;
    background-color: #DFDDDD;
}

.comment-container .btn-submit {
    width: 180rpx;
    height: 70rpx;
    margin: 0 20rpx;
    font-size: 36rpx;
    font-family: SourceHanSansCN-Medium;
    font-weight: 500;
    color: rgba(130, 129, 129, 1);
    line-height: 70rpx;
```

```
}
```

底部评论框也是 `Flex` 布局的应用，这里不再进行分析，读者自己分析即可。

最后我们来看下图书详情页面的业务逻辑处理：

```
// pages/detail/detail.js
import config from '../../config/config.js';
import data from '../../utils/data.js';

Page({
    data: {
        id: '',
        bookInfo: {},
        favoriteList: [],
        isMarked: false,
        commentList: [],
        currentComment: '',
        userInfo: null,
    },
    onLoad: function (query) {
        //query 是通过 url 传过来的参数的集合
        if (query.id && query.name) { //处理是否有 id 和 name
            //匹配模拟数据
            let bookInfo = {};
            if (query.id.indexOf('android_') != -1) {
                bookInfo = data.detail.androidBookInfo;
            } else if (query.id.indexOf('ios_') != -1) {
                bookInfo = data.detail.iosBookInfo;
            } else if (query.id.indexOf('fe_') != -1) {
                bookInfo = data.detail.feBookInfo;
            } else if (query.id.indexOf('backend_') != -1) {
                bookInfo = data.detail.backendBookInfo;
            } else if (query.id.indexOf('ai_') != -1) {
                bookInfo = th.data.aiBookInfo;
            }
            bookInfo.id = query.id;
            bookInfo.name = query.name;
            this.setData({
                id: query.id,
                bookInfo: bookInfo
            });
        }
        //读取已经收藏的图书列表
        let favoriteBooks =
wx.getStorageSync(config.cacheKey.favoriteBooks);
        if (favoriteBooks) {
```

```
            this.setData({
                favoriteList: favoriteBooks
            });
            this.setData({ //更新收藏状态
                isMarked: this.isMarked()
            });
        }
        //读取评论列表
        let commentList = wx.getStorageSync(this.data.id);
        if (commentList) {
            this.setData({
                commentList: commentList
            });
        }
        let userInfo = wx.getStorageSync(config.cacheKey.userInfo);
        if (userInfo) {
            this.setData({
                userInfo: userInfo
            });
        }
    },
    toFavorite: function () { //收藏按钮点击
        if (this.isMarked()) { //图书已被收藏，则取消收藏
            let favoriteList = this.data.favoriteList;
            // indexOf 返回已经收藏的图书在 list 中的下标
            // splice(0,1); splice 函数有两个参数，第一个表示从哪个坐标开始
删除，第二个表示删除多少个

 favoriteList.splice(favoriteList.indexOf(this.data.bookInfo), 1);
            // 更新收藏列表
            wx.setStorageSync(config.cacheKey.favoriteBooks,
favoriteList);
            this.setData({
                favoriteList: favoriteList,
                isMarked: this.isMarked()
            });
        } else { // 图书未被收藏，则执行收藏
            let favoriteList = this.data.favoriteList;
            // push(); push 函数的作用是将元素追加到数组末尾
            favoriteList.push(this.data.bookInfo);
            // 更新收藏列表
            wx.setStorageSync(config.cacheKey.favoriteBooks,
favoriteList);
            this.setData({
                favoriteList: favoriteList,
                isMarked: this.isMarked()
            });
```

```
        }
    },
    isMarked: function () { //判断当前图书有没有被收藏过
        for (let book of this.data.favoriteList) {
            if (book.id == this.data.id) {
                return true;
            }
        }
        return false;
    },
    onCommentInput: function (e) { //当评论输入框输入内容时回调
        this.setData({
            currentComment: e.detail.value
        });
    },
    toSubmitComment: function () { //保存评论
        if (!this.data.currentComment) return;
        let comment = {
            username: this.data.userInfo.username,
            content: this.data.currentComment
        };
        let commentList = wx.getStorageSync(this.data.id);
        if (!commentList) {
            commentList = [];
        }
        commentList.push(comment);
        this.setData({
            commentList: commentList
        });
        wx.setStorageSync(this.data.id, commentList);
        this.setData({
            currentComment: ''
        });
    }
})
```

7.2.6 收藏页面

接下来我们来实现商城的最后一个页面图书收藏页面，先看下效果图：

图7-2-6 收藏页面

通过页面效果图可以看到，收藏页面只有一个滑动列表。我们来看下页面结构图：

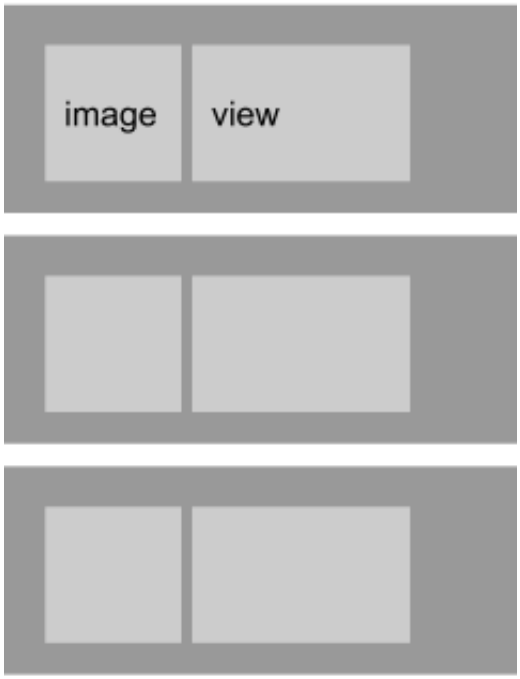

图7-2-6 收藏页面结构图

收藏页面整体比较简单，只需要对其中一个 `item` 进行布局就行了。我们先来看下收藏页面的结构代码：

```
<!-- pages/favorite/favorite.wxml -->
<view>
  <view class="list-container">
      <!-- 使用 for 循环遍历生成 item 的 view -->
      <view class="item"
            wx:for="{{dataList}}"
            data-item="{{item}}"
            wx:key="index"
            bindtap="onItemClick">
          <image class="cover" src="{{item.image}}" aspectFill />
          <view class="title">{{item.name}}</view>
      </view>
  </view>
</view>
```

item 的布局也是 `Flex` 布局的应用，对大家来说已经很熟悉了，我们来看下具体实现：

```
/* pages/favorite/favorite.wxss */
```

```
.list-container {
    width: 100%;
    padding: 40rpx 0;
}

.list-container .item {
    width: 100%;
    height: 200rpx;
    padding: 0 80rpx;
    line-height: 200rpx;
    margin-top: 10rpx;
    background-color: #ffffff;
    /* 使用 flex 布局 */
    display: flex;
    /* 方向为纵向 */
    flex-direction: row;
    /* 设置为不可换行 */
    flex-wrap: nowrap;
    /* 从左边开始布局 */
    justify-content: flex-start;
    /* 垂直居中 */
    align-content: center;
    align-items: center;
}

.list-container .item .cover {
    width: 160rpx;
    height: 160rpx;
}

.list-container .item .title {
    margin-left: 20rpx;
}
```

最后我们来看下收藏页面的业务逻辑处理：

```
// pages/favorite/favorite.js
import config from '../../config/config.js';

Page({
    data: {
        dataList: []
    },
    onLoad: function (query) {
        //读取收藏的图书列表
```

```
        let favoriteBooks =
wx.getStorageSync(config.cacheKey.favoriteBooks);
        this.setData({
            dataList: favoriteBooks
        });
    },
    onItemClick: function (e) {//图书点击跳转到详情页
        let item = e.currentTarget.dataset.item;
        wx.navigateTo({
            url: "/pages/detail/detail?id=" + item.id + "&name=" +
item.name
        });
    }
})
```

7.3 打包上线

这一章节我们介绍下，怎么打包上线我们的开发的小程序。

7.3.1 上传代码

项目开发完成后就可以上传代码了，在微信开发者工具右上角可以找到 `上传` 按钮，点击上传然后再点击确定。

图7-3-1 微信开发者工具中的上传

下面需要填写上传的版本信息，填写相关信息后点击上传即可。

图7-3-2 填写上传信息

7.3.2 提交审核

代码上传后，我们需要登录到微信公众平台。在微信公众平台的左侧找到 `开发管理` 。

图7-3-3 开发管理

点击开发管理之后，我们在底部可以看到刚才上传的代码。

图7-3-4 提交审核

点击提交审核，然后会看到一些条款不管它，打上对勾点击下一步。

图7-3-5 确认提交审核

到这里需要我们填写一些信息，首先需要选中首页页面路径，也就是 `app.json` 中 `pages` 下的第一个路径。

然后填写完信息点击提交审核。

图7-3-6 填写项目配置

提交审核之后会在 `开发管理` 下看到小程序已经处于审核中的状态了。

图7-3-7 已提交的审核版本

一般 2-3 小时就会审核通过了，审核通过之后，管理员的微信中会收到小程序通过审核的通知。此时登录 `小程序管理后台` - `开发管理` - `审核版本中` 可以看到通过审核的版本。然后需要我们点击进行公测，最后点击发布，即可发布小程序，发布后就能在微信中搜索到我们的小程序了。