《微信小程序综合实践》实验报告五

小程序实战 —— 音乐小程序制作

姓名_____ 专　　业_____

学号_____ 联系方式_____

一、实验目的

1. 通过本次实验综合运用网页前端知识，完成音乐小程序设计制作全过程；
2. 掌握使用 wxml 对小程序进行基本页面布局的基本方法；
3. 熟练使用 wxss 对小程序的页面进行合理布局；
4. 掌握小程序生命周期函数的用法，能使用 JavaScript 在小程序页面加载前对资源进行初始化；
5. 掌握模块化的基本技能，能进行组件复用；
6. 掌握小程序页面基本属性的配置方法，能对页面进行个性化设置；
7. 掌握小程序静态数据加载的方法；

二、实验内容及要求

准备工作：为了方便后续的学习和实践，建议同学们自带电脑，在自己电脑上进行实验内容，并完成实验报告。本次实验为基础内容，认真完成本次实验内容即可掌握开发微信小程序的基本工具使用。

（一） 实验内容

1. 对音乐小程序进行功能分析，找到小程序编写入手点；
2. 对页面数据进行渲染，并对相应的布局设置样式；
3. 将相同逻辑抽象成组件，通过父子组件传值和循环渲染完成复杂页面；
4. 自定义数据配置，通过引入本地自定义数据配置，对页面进行渲染；

（二） 实验步骤（仔细阅读，按照步骤完成实验）

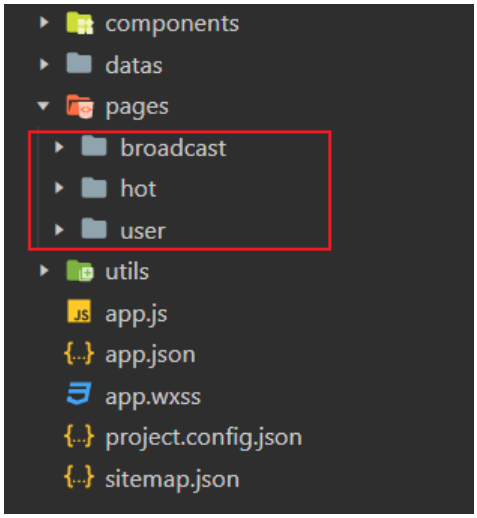
（本次实验用到的静态资源文件可见学在浙大上的附件）

1. 对音乐小程序进行功能分析，为实际页面编写做准备

（1） 查看音乐小程序效果图。

（2） 对效果图进行分析，确定小程序的页面结构。

通过效果图可以看出，该小程序主要包含三个页面，在 pages 目录下分别新建三个页面对该页面进行编写：



（3） 完成页面的基本设置，为页面开发做准备。

- 在 app.json 中 pages 数组中对三个页面进行注册；
- 在 app.json 中 tabBar 数组中定义导航栏；

2. 编写「热歌推荐」页面

（1） 首先确保点击 tabBar 中按钮，三个页面能够正常进行跳转；

（2） 分析页面数据结构。

通过观察，页面包括三位歌手、他们的代表作歌曲和图片封面，首先编写周杰伦相关的内容

（3） 在 hot.js 文件中定义好所需要的数据，如下图所示：

```
data: {
  id: 1,
  name: "周杰伦",
  src: "http://cdnmusic.migu.cn/picture/2019/1031/0254/ASd6c2d9697d2a4f5f96508c8a7ec8b1a8.jpg",
  songs: [
      { id: 1, title: "七里香"},
      { id: 2, title: "说好不哭"},
      { id: 3, title: "梯田"}
  ]
},
```

（4） 编写 hot.wxml 文件，先不考虑样式，使数据能够正常显示，可参考下图：

```
<view>
    <view class="author">{{name}}</view>
    <view class="content">
        <view class="song-list">
            <view class="song" wx:for="{{songs}}" wx:key="index" wx:for-item="song">
                <text>{{song.id}}--</text>
                <text>{{song.title}}</text>
            </view>
        </view>
        <view class="profile">
            <image class="banner" mode="aspectFit" src="{{src}}"></image>
        </view>
    </view>
</view>
```

编写完成后，此时的「热歌推荐」页面应该如下图所示：

3. 为「热歌推荐」页面添加样式

（1） 给歌手增加样式：

```
.author {
  font-weight: bold;
  font-size: 40rpx;
  background-color: #f5f5f5;
}
```

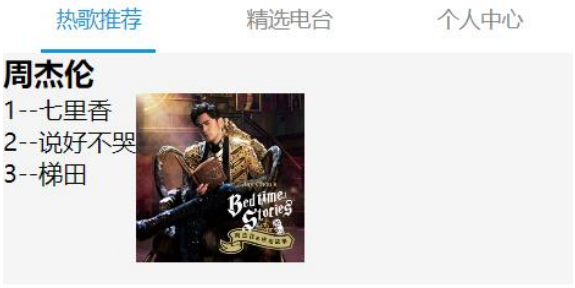
（2） 调整歌手封面大小

```
.profile {
  width: 250rpx;
  height: 250rpx;
}

.banner {
  width: 220rpx;
  height: 220rpx;
}
```

（3） 使用弹性布局（flex 布局），使封面显示在歌曲的右侧：

在 .content 中添加： `display: flex;` 使整个容器都采用弹性布局。

完成后的效果应该如下图所示：



（4） 使封面显示在最右侧并对齐

在 .profile 中添加属性 ： `margin-left: auto;`

完成后的效果显示封面已对齐在了最右侧：

（5） 调整歌手姓名框和内容框的大小及边距，让页面更有层次感。

（6） 为页面添加背景色

在 hot.wxss 最上方添加：

```
page{background-color: darkslategrey;}
```

此时效果如下：

（7） 使歌曲名填充整个空间

为歌曲名 .song 添加样式：

```
.song {
  padding: 20rpx;
}
```

完成后的效果如下：

（8） 让封面照片显示在上下边距的正中间

为 .profile 增加样式：

```
display: flex;
align-items: center;
justify-content: center;
```

完成后的效果如下图所示：

（9） 为边框增加圆角，使得卡片更加美观

分别为 .author 和 .content 添加样式：

```
border-radius: 5px;
```

完成效果如图：

（10） 调整歌手名左边距

为 .author 添加样式：

```
padding-left: 10px;
```

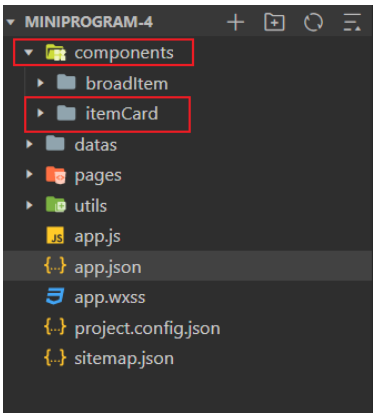
效果如下：

4. 实现组件复用，将相同逻辑抽象成一个组件并引入

（1） 将周杰伦及其歌曲和封面抽象成组件

观察到其余歌手的显示方式与周杰伦的相同，应该将刚才的内容抽象成为一个可复用的组件模板，对数据循环渲染。

（2） 新建 components 文件夹，并在其下新建 itemCard 组件（components 文件夹创建好之后，右键新建 itemCard 文件夹，在 itemCard 文件夹上右键新建 Component 并命名为 itemCard）：



（3） 将原来 hot.wxml 和 hot.wxss 中的内容拷贝到 itemCard.wxml 和 itemCard.wxss 中（除了 `page{background-color:darkslategrey;}`）

（4） 删掉原来 hot.wxml 和 hot.wxss 中的内容，并引入 itemCard 这个组件：在 hot.json 中配置引入该组件：

```
{
  "usingComponents": {
    "itemCard": "../../components/itemCard/itemCard"
  }
}
```

（5） 在 hot.wxml 中引入该组件

```
<view class="music-hot">
    <itemCard />
</view>
```

但此时因为没有传递数据，所以显示结果如下图所示：

（6） 为 itemCard 组件传递数据。

a) 修改 hot.js 中的数据，将它们写在一个对象中。（注意：musics 对象还是要放在 data 对象里面，此处截图没有截到）

```
musics: {
  id: 1,
  name: "周杰伦",
  src: "http://cdnmusic.migu.cn/picture/2019/1031/0254/ASd6c2d9697d2a4f5f96508c8a7ec8b1a8.jpg",
  songs: [
    { id: 1, title: "七里香"},
    { id: 2, title: "说好不哭"},
    { id: 3, title: "梯田"}
  ]
}
```

b) 在 hot.wxml 中引入该数据

```
<view class="music-hot">
    <itemCard item="{{musics}}" />
</view>
```

c) 配置 itemCard，使它能够接受到该值：在 itemCard.js 中设置 propertities：

```
// components/itemCard.js
Component({
  /**
   * 组件的属性列表
   */
  properties: {
    item: {
      type: Object,
      value: {}
    },
  },
```

d) 此时还无法正常显示，还需要对 itemCard.wxml 再进行修改

```
<view>
  <view class="author">{{item.name}}</view>
  <view class="content">
    <view class="song-list">
      <view class="song" wx:for="{{item.songs}}" wx:key="index" wx:for-item="song">
        <text >{{song.id}} -- </text>
        <text >{{song.title}}</text>
      </view>
    </view>
    <view class="profile">
      <image class="banner" mode="aspectFit" src="{{item.src}}"/>
    </view>
  </view>
</view>
```

（7） 查看显示结果

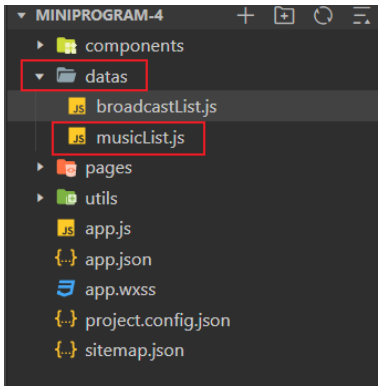
5. 引入自定义数据，使用生命周期函数加载自定义数据，并通过自定义数据进行渲染

（1） 回顾之前的数据获取方式

之前的数据获取方式将数据集定义在 hot.js 中的 data 对象中，当数据量很大或者需要修改时，hot.js 维护将变的十分不方便。

（2） 新建配置文件，将数据写在该配置文件中

a) 新建 datas 文件夹，并在其下建立 musicList.js 文件。



b) 在 musicList 文件中将数据写入一个对象：

```
var json = [
    {
        "id" : 1,
        "name": "周杰伦",
        "src": "http://cdnmusic.migu.cn/picture/2019/1031/0254/ASd6c2d9697d2a4f5f96508c8a7ec8b1a8.jpg",
        "songs": [
            { "id": 1, "title": "七里香"},
            { "id": 2, "title": "说好不哭"},
            { "id": 3, "title": "梯田"}
        ]
    },
    {
        "id": 2,
        "name": "李荣浩",
        "src": "http://cdnmusic.migu.cn/picture/2019/1031/0130/AM5251251132424657bd7190cc3690fa40.jpg",
        "songs": [
            { "id": 1, "title": "太坦白"},
            { "id": 2, "title": "不将就"},
            { "id": 3, "title": "老街"}
        ]
    },
    {
        "id" : 3,
        "name": "许嵩",
        "src": "http://cdnmusic.migu.cn/picture/2019/0422/0849/AS1704070955546876.jpg",
        "songs": [
            { "id": 1, "title": "有何不可"},
            { "id": 2, "title": "雅俗共赏"},
            { "id": 3, "title": "素颜"}
        ]
    }
]

module.exports = {
```

c) 将该对象导出，使得外部可以引入该数据

在 musicList.js 文件末尾添加：

```
module.exports = {
...
    musics: json
}
```

则外部文件便可通过 musics 引入该数据。

（3） 在 hot.js 中引入刚定义好的数据

a) 在 hot.js 开头添加引入语句：

```
import musicList from "../../datas/musicList.js";
```

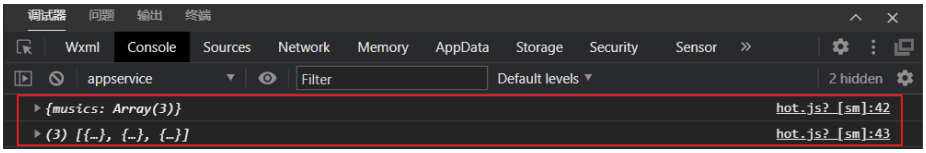
b) 将原来 data 中 musics 的值删去；

c) 在生命周期函数 onLoad 中更新 data 数据：

```
/**
 * 生命周期函数--监听页面加载
 */
onLoad: function (options) {
  this.setData({
    musics: musicList.musics
  })
},
```

d) 在生命周期函数 onLoad 中添加辅助信息，查看是否正确获取到数据：

```
onLoad: function (options) {
  this.setData({
    musics: musicList.musics
  })
  console.log(musicList);
  console.log(musicList.musics);
},
```



e) 此时还不能正确显示，因为此时返回的数据是一个数组，需要对 hot.wxml 进行修改。

```
<view class="music-hot">
    <view wx:for="{{musics}}" wx:for-item="item" wx:for-index="index" wx:key="index">
        <itemCard item="{{item}}"/>
    </view>
</view>
```

修改之后的显示结果如下图所示：

f) 为顶部添加边距：

```
<view style="padding-bottom: 20px"></view>
<view class="music-hot">
    <view wx:for="{{musics}}" wx:for-item="item" wx:for-index="index" wx:key="index">
        <itemCard item="{{item}}"/>
    </view>
</view>
```

添加之后的显示效果如下图所示：

6. 编写「精选电台」界面

（1） 准备数据源：

在 datas 文件夹下新建 broadcastList.js 文件，在其中写入所需要的配置，并在最后对数据进行导出：

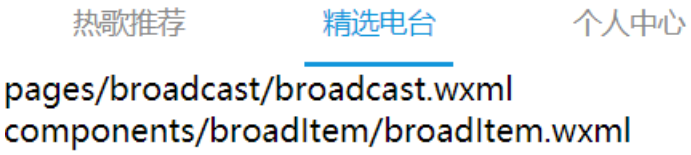
```
var json = [
    {
        name: "个性电台",
        thumb: "https://y.qq.com/music/common/upload/t_music_radio/1261958.jpg?max_age=2592000"
    },
    {
        name: "随心听",
        thumb: "https://y.qq.com/music/common/upload/t_music_radio/1260297.jpg?max_age=2592000"
    },
    {
        name: "经典",
        thumb: "https://y.qq.com/music/common/upload/t_music_radio/2054444.jpg?max_age=2592000"
    },
    {
        name: "网络流行",
        thumb: "https://y.qq.com/music/common/upload/t_music_radio/2054505.jpg?max_age=2592000"
    },

    {
        name: "抖音神曲",
        thumb: "https://y.qq.com/music/common/upload/t_music_radio/2054458.jpg?max_age=2592000"
    },
    {
        name: "深度催眠",
        thumb: "https://y.qq.com/music/common/upload/t_music_radio/3188826.jpg?max_age=2592000"
    },
    {
        name: "情感治愈站",
        thumb: "https://y.qq.com/music/common/upload/t_music_radio/2929760.jpg?max_age=2592000"
    },
    {
        name: "KTV必点歌",
        thumb: "https://y.qq.com/music/common/upload/t_music_radio/2054525.jpg?max_age=2592000"
    },
    {
        name: "精选招牌歌",
        thumb: "https://y.qq.com/music/common/upload/t_music_radio/2054526.jpg?max_age=2592000"
    },
    {
        name: "热门翻唱",
        thumb: "https://y.qq.com/music/common/upload/t_music_radio/2054491.jpg?max_age=2592000"
    }
]

module.exports = {
    broadcasts: json
}
```

（2） 新建 broadItem 组件，并在 broadcast 页面中进行引入：

```
<view>pages/broadcast/broadcast.wxml</view>
<view class="page">
    <broadItem item="{{item}}"></broadItem>
</view>
```

```
{
  "usingComponents": {
    "broadItem": "../../components/broadItem/broadItem"
  }
}
```

当页面能正常显示出 broadcast 和 broadItem 时说明引入成功：



pages/broadcast/broadcast.wxml
components/broadItem/broadItem.wxml

（3） 引入电台数据：

在 broadcast.js 中引入数据源，并在说明周期函数中对数据进行初始化：

```
import broadcastList from "../../datas/broadcastList.js";
Page({
  /**
   * 页面的初始数据
   */
  data: {
    broadcasts: []
  },
  /**
   * 生命周期函数--监听页面加载
   */
  onLoad: function (options) {
    this.setData({
      broadcasts: broadcastList.broadcasts
    });
  },
```

（4） 在 broadcast.wxml 中循环渲染 broadItem：

```
<view class="page">
    <view class="content" wx:for="{{broadcasts}}" wx:for-item="item" wx:for-index="index" wx:key="index">
        <broadItem item="{{item}}"></broadItem>
    </view>
</view>
```

当页面显示如下时，说明正常引入数据和组件，并可正常渲染。

（5） 开始编写 broadItem 组件：

a) 编写 broadItem.wxml 文件：

```
<view class='broad-item' style="background-image: url('{{item.thumb}}');" >
    <text class="broad-text">{{item.name}}</text>
</view>
```

b) 设置 broadItem.js 文件，接收父组件传递的属性：

```
Component({
  /**
   * 组件的属性列表
   */
  properties: {
    item: {
      type: Object,
      value: {}
    }
  },
```

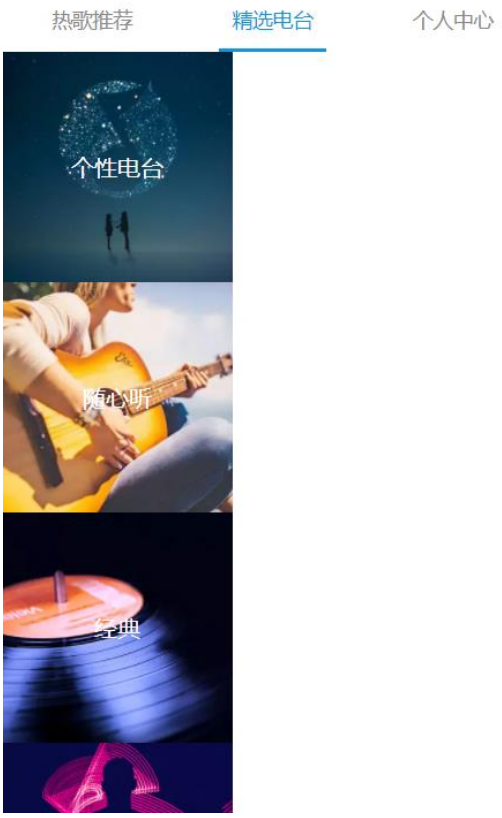
c) 查看显示结果：

（6） 为 broadItem 组件配置样式：

a) 设置电台封面图大小和文字样式：

```
/* components/broadItem/broadItem.wxss */
.broad-item{
  width: 320rpx;
  height: 320rpx;
  display: flex;
  justify-content: center;
  align-content: center;
  background-size: 100% 100%;
  position: relative;
  color: white;
  display: flex;
  align-items: center;
  z-index: 1;
}

.broad-text {
  color: white;
  display: flex;
  align-items: center;
  z-index: 1;
}

/* .broad-item::after{
```

查看效果如下图所示：



b) 为了使得封面文字能更清楚显示，继续调整给图片增加阴影效果：

```
.broad-item::after{
  position: absolute;
  top: 0;
  left: 0;
  content: '';
  width: 100%;
  height: 100%;
  background-color: rgba(0, 0, 0, .5);
  z-index: 0;
}
```

修改之后效果如下：

c) 调整外部容器，使得封面图片能够每行两个对齐显示

为父组件 broadcast 添加样式，在 broadcast.wxss 中添加样式：

```
.page{
  background-color: #fafafa;
  width: 700rpx;
  margin: 0 auto;
  padding-top: 40rpx;
  padding-bottom: 110rpx;
}

.page::after{
  display: block;
  clear: both;
}
.content {
    width: 350rpx;
    height: 350rpx;
    float: left;
    display: flex;
    justify-content: center;
    align-items: center;
}
```

调整之后的截图如下：

作业上交内容与事项：

1．实验报告文档：按照要求完成实验并将关键步骤实验结果进行截图记录。注意文档工整；

2．程序代码：如果有程序代码或其他相关材料，可汇总压缩所有文件并上传压缩包。注意文件命名；

本期实验作业上期限：

请在实验设置的截止日期内提交实验报告，若逾期提交，成绩会适当被打折。

本次作业上交内容：

- 实验报告文档(.docx)
- 程序代码压缩文档(.zip)

三、实验感受及记录

（一）实验感受（本次实验遇到的问题、主要收获等内容）

（二）实验记录（实验过程中关键步骤截图记录及文字描述）

2.1 独立完成实验步骤的内容，将所有示例进行复现（不允许重复，要有适当改动）。

2.2 实验中三个页面中「个人中心」页面还没有完成，请发挥创造力在完成作业 2.1 的基础上，对「个人中心」页面进行编写。

2.3 将完成作业 2.2 后的源代码文件压缩上传提交，并在下方附上三个页面的截图。