

## 《微信小程序综合实践》实验报告七

### 小程序实战 —— 音乐小程序在线播放与云开发

姓名 张博文 专业 信息工程

学号 3220104331 联系方式 18755860974

#### 一、实验目的

1. 在上一次实验的基础上，对小程序进行完善，通过本次实验得到功能完备的小程序并可发布上线；
2. 综合运用小程序所提供的多媒体组件与事件机制，实现音乐小程序核心功能；
3. 掌握父子组件、兄弟组件之间传值的方法，掌握全局数据的用法；
4. 掌握小程序状态同步方法，通过全局数据同步相同组件不同实例之间的数据；
5. 掌握云开发基本环境的配置方法，可以将普通环境增强为云开发环境；
6. 使用云开发提供的接口和云函数，对有用信息持久化存储到云数据库；
7. 熟练掌握对云数据库数据增加、修改、删除、查找等操作，通过云数据库获取数据并对组件状态进行更新；

#### 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. 在上一次实验的基础上，使用 `audio` 组件为音乐小程序添加歌曲播放器；
2. 将歌曲播放器固定在底部栏并在每个页面进行显示；
3. 自定义适合屏幕尺寸的播放器组件；
4. 对播放器歌曲信息进行同步，保证页面切换时当前播放信息不会更改；
5. 添加音乐详情页，能点击音乐详情页进行歌曲播放；
6. 使用云开发提供收藏歌曲功能，保存音乐收藏信息；

## (二) 实验步骤（仔细阅读，按照步骤完成实验）

（本次实验所需的静态资源文件可见学在浙大上的附件）

### 1. 为音乐小程序添加播放器功能。

(1) 查看官网多媒体组件内容，寻找播放器相关组件。

<https://developers.weixin.qq.com/miniprogram/dev/component/audio.html>

在热歌推荐的 wxml 文件中引入多媒体组件 audio:

```
<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>
  <audio id="myAudio" class="player"
    poster="{{poster}}" name="{{name}}" author="{{author}}" src="{{src}}"
    controls loop>
  </audio>
</view>
```

(2) 在 hot.js 文件中为 audio 组件添加相关数据，测试组件功能，下图以七里香为例：

```
poster: 'https://y.gtimg.cn/music/photo_new/T002R300x300M000003DFRzD192KKD.jpg?max_age=2592000',
name: '七里香',
author: '周杰伦',
src: 'https://mp3.9ku.com/hot/2004/08-03/58714.mp3',
```

(3) 看「热歌推荐」页面显示效果，并测试能否正常播放音乐：



### 2. 修改播放器位置固定在底部栏

(1) 给播放器添加样式，使播放器始终在页面中显示：

- i. 使用绝对定位，将播放器固定在页面底部：

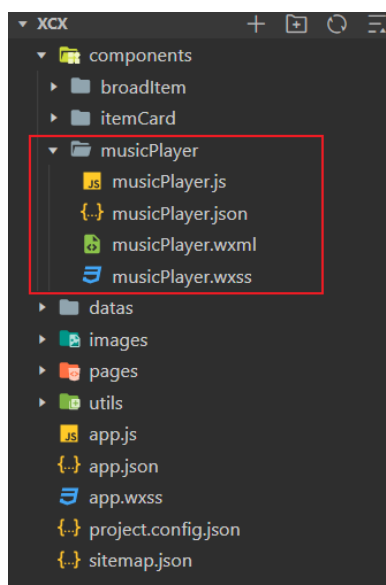
```
.player {  
  position: fixed;  
  bottom: 0;  
}
```

- ii. 查看页面显示效果，不管页面如何滚动，播放器应该始终显示在页面底部：



- (2) 为精选电台和个人中心也添加播放器：

- i. 新建组件 musicPlayer



ii. 在 musicPlayer.wxml 文件中写入代码：

```

1 <view>
2   <audio id="myAudio" class="player"
3     poster="{{song.poster}}" name="{{song.name}}" author="{{song.author}}" src="{{song.src}}"
4     controls loop>
5   </audio>
6 </view>

```

iii. 在 musicPlayer.wxss 文件中写入如下代码：

```

1 /* components/musicPlayer/musicPlayer.wxss.wxss */
2 .player {
3   position: fixed;
4   bottom: 0;
5 }

```

iv. 在 musicPlayer.js 文件中添加组件属性：

```

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

```

v. 以「热歌推荐」为例，为 musicPlayer 提供数据并引入该组件：

```

// poster: 'https://y.gtimg.cn/music/photo_new/T002R300x300M000003DFRzD192KKD.jpg?max_age=2592000',
// name: '七里香',
// author: '周杰伦',
// src: 'https://mp3.9ku.com/hot/2004/08-03/58714.mp3',
song: {
  poster: 'https://y.gtimg.cn/music/photo_new/T002R300x300M000003DFRzD192KKD.jpg?max_age=2592000',
  name: '七里香',
  author: '周杰伦',
  src: 'https://mp3.9ku.com/hot/2004/08-03/58714.mp3',
}

```

```

<!-- <audio id="myAudio" class="player"
  poster="{{poster}}" name="{{name}}" author="{{author}}" src="{{src}}"
  controls loop>
</audio> -->

<musicPlayer song="{{song}}"></musicPlayer>

```

```

<usingComponents> {
  "itemCard": "../components/itemCard/itemCard",
  "musicPlayer": "../components/musicPlayer/musicPlayer"
}

```

### 3. 自定义播放器组件

- (1) 微信小程序官方提供的 `audio` 组件无法自定义修改样式且该组件已不在维护。
- (2) 编写自定义播放器的 `wxml`，包含所要显示的内容：

```
<view class="player" bindtap="handleMusic">
  <view class="p-left">
    <image class="music-pic paused" src="{{song.poster}}" mode='aspectFit'></image>
    <view class="music-info">
      <text class="music-name">{{song.name}}</text>
      <text class="music-artist">{{song.author}}</text>
    </view>
  </view>
  <view class="p-right"></view>
</view>
```

- (3) 修改 `player` 类样式，确定播放器的位置和大小：

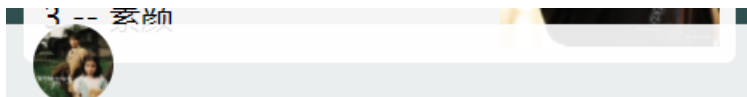
```
/* components/musicPlayer/musicPlayer.wxss.wxss */
.player {
  position: fixed;
  bottom: 0;
  width: 100%;
  height: 80px;
  background-color: rgba(255, 255, 255, 0.9);
}
```

完成之后如下图所示：



- (4) 为歌曲封面、歌曲名和歌手设置样式，使得能够正常显示：
  - i. 为歌曲封面设置样式：

```
.music-pic {
  width: 80rpx;
  height: 80rpx;
  border-radius: 50rpx;
  margin-left: 30rpx;
}
```



- ii. 此时歌手和歌曲名在下方未显示出来，为外层容器设置 flex 布局，使得他们也能显示在屏幕上：

```
.p-left {
  display: flex;
}
```



- iii. 使歌曲和歌手从上到下排列：

```
.music-info {
  display: flex;
  flex-direction: column;
  justify-content: space-around;
  margin-left: 20rpx;
}
```



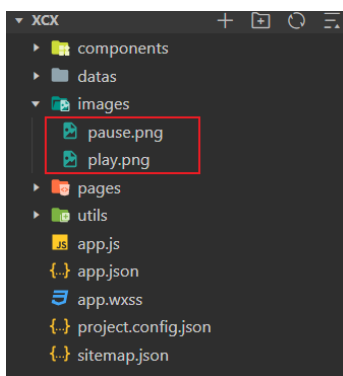
- iv. 修改歌手和歌曲名字体样式：

```
.music-name {
  font-size: 30rpx;
  font-weight: bold;
}

.music-artist {
  font-size: 25rpx;
  color: #6a6a6a;
  font-weight: bold;
}
```



- v. 根据之前实验报告，添加暂停和播放图标并引入：



```
<view class="player" bindtap="handleMusic">
  <view class="p-left">
    <image class="music-pic paused" src="{{song.poster}}" mode='aspectFit'></image>
    <view class="music-info">
      <text class="music-name">{{song.name}}</text>
      <text class="music-artist">{{song.author}}</text>
    </view>
  </view>
  <view class="p-right">
    <image src="{{playState == 1?'/images/pause.png':'/images/play.png'}}" mode='aspectFit'></image>
  </view>
</view>
```

- vi. 在组件的初始数据中为播放状态设置初始值：

```
/**
 * 组件的初始数据
 */
data: {
  playState: 1,
},
```

- vii. 此时，播放状态图标还需要设置样式才能正确显示，为 player 类新添加属性：

```
.player {
  position: fixed;
  bottom: 0;
  width: 100%;
  height: 80rpx;
  background-color: rgba(255, 255, 255, 0.9);
  padding: 10rpx 0;
  display: flex;
  justify-content: space-between;
}
```

- viii. 修改暂停/播放按钮的大小和位置：

```

.p-right {
  align-self: center;
  margin-right: 30rpx;
}

.p-right image {
  width: 100rpx;
  height: 100rpx;
}

```

保存后效果如下图：



将 js 中的 playState 改为 0，保存后应该能看到切换到了播放按钮。

(5) 使播放器组件能够播放音乐：

添加时间处理函数，使得点击播放按钮后播放器能够播放音乐。通过分析得知，播放器是音乐小程序的核心功能，每个页面都需要播放器组件，所以应该将播放器组件中播放的歌曲信息作为全局数据。

- i. 使用小程序 API 在 app.js 中注册一个音乐播放器：

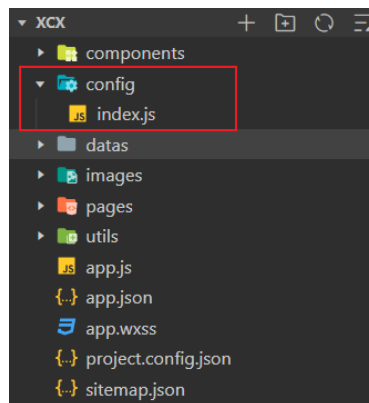


```
globalData: {
  userInfo: null,
  audio: wx.createInnerAudioContext()
}
```

ii. 添加小程序加载时的默认播放歌曲信息:

```
globalData: {
  userInfo: null,
  audio: wx.createInnerAudioContext(),
  playState: DEFAULT_MUSIC.DEFAULT_MUSIC.playState,
  musicPic: DEFAULT_MUSIC.DEFAULT_MUSIC.musicPic,
  musicName: DEFAULT_MUSIC.DEFAULT_MUSIC.musicName,
  musicUrl: DEFAULT_MUSIC.DEFAULT_MUSIC.musicUrl,
  artistName: DEFAULT_MUSIC.DEFAULT_MUSIC.artistName,
}
```

iii. 编写配置文件, 添加 DEFAULT\_MUSIC 信息:



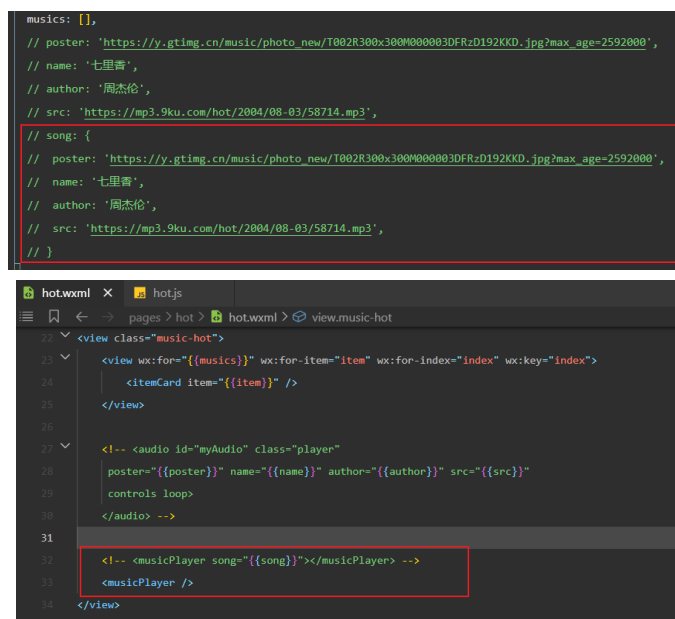
```
//歌曲默认信息配置
const DEFAULT_MUSIC = {
  musicName: '七里香',
  artistName: '周杰伦',
  musicPic: "https://y.gtimg.cn/music/photo_new/T002R300x300M000003DFRzD192KKD.jpg?max_age=2592000",
  musicUrl: "https://mp3.9ku.com/hot/2004/08-03/58714.mp3",
  playState: 0
}

module.exports = {
  ...
  DEFAULT_MUSIC
}
```

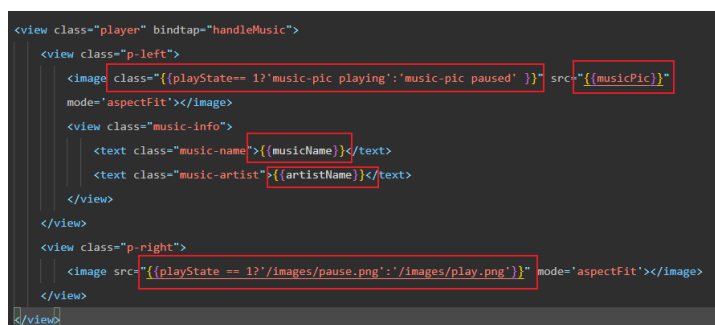
iv. 在 app.js 中引入 DEFAULT\_MUSIC:

```
//app.js
import DEFAULT_MUSIC from './config/index';
App({
  onLaunch: function () {
    // 展示本地存储能力
    var logs = wx.getStorageSync('logs') || []
    logs.unshift(Date.now())
    wx.setStorageSync('logs', logs)
  }
})
```

- v. 删掉「热歌推荐」页面中为组件传递的数据：



- vi. 修改 musicPlayer 组件的数据和数据传递方式（需要在 musicPlayer.js 中声明 App 实例获取 globalData 并赋值给下图的模板变量），同时修改 image 表情的样式，使其可根据 playState 的值进行调整：



其中 musicPlayer.js 中声明 App 实例的方式如下：

```

2 var app = getApp()
3 Component({

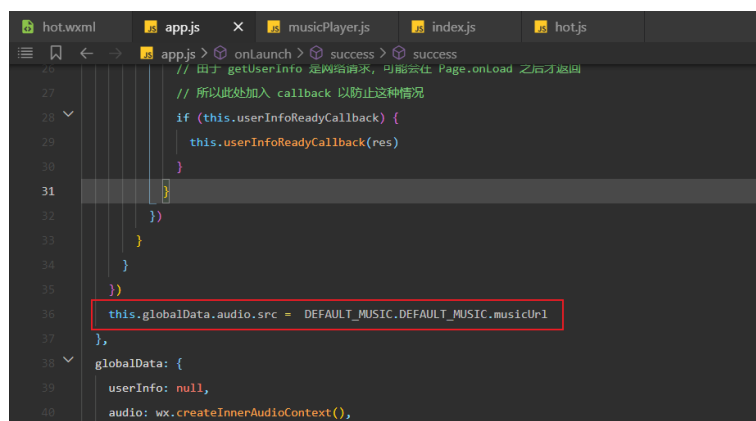
```

然后需要用 globalData 给 musicName、artistName、musicPic 赋值，如图所示：

```

data: {
  playState: 1,
  musicPic: app.globalData.musicPic,
  musicName: app.globalData.musicName,
  artistName: app.globalData.artistName,
},

```



保存之后播放器组件应该显示正常：

- (6) 为 musicPlayer 添加播放和暂停事件，使得能够播放音乐：  
通过上述变量我们约定全局变量 playState 为 0 时暂停音乐，为 1 时播放音乐，并且通过 musicPlayer.wxml 约定点击事件为 handleMusic

- i. 准备工作：为播放器设置初始播放源。在 app.js 的生命周期函数中添加：

ii. 在 musicPlayer.js 中编写 handleMusic:

此时播放音乐的函数为 play(), 暂停音乐的函数为 pause()。

iii. 编写播放音乐的函数和暂停音乐的函数:

```
play: function () { // 播放音乐
  app.globalData.audio.play();
  app.globalData.playState = 1;
},
pause: function () { // 暂停播放
  app.globalData.audio.pause();
  app.globalData.playState = 0;
},
```

```
/**
 * 组件的方法列表
 */
methods: {
  handleMusic: function () {
    console.log(app.globalData);
    switch (this.data.playState) {
      case 0:
        this.setData({ // 点击后由0变为1, 播放音乐
          playState: 1
        });
        this.play();
        break;
      case 1:
        this.setData({ // 点击后由1变为0, 暂停播放
          playState: 0
        });
        this.pause();
        break;
    }
  }
}
```

iv. 此时, 点击播放按钮应该能正确播放/暂停音乐。

(7) 为播放器组件在播放音乐时, 歌曲封面添加旋转效果

在 musicPlayer.wxss 中添加 playing 和 pause 类的样式：

```
.playing {  
  animation: playing 3.5s linear infinite;  
  animation-play-state: running;  
}  
  
.paused{  
  animation: playing 3.5s linear infinite;  
  animation-play-state: paused;  
}  
  
@keyframes playing{  
  0%{-webkit-transform:rotate(0deg);}  
  25%{-webkit-transform:rotate(90deg);}  
  50%{-webkit-transform:rotate(180deg);}  
  75%{-webkit-transform:rotate(270deg);}  
  100%{-webkit-transform:rotate(360deg);}  
}
```

保存后播放音乐，将看到歌曲封面在旋转，暂停后旋转停止。

#### 4. 增加音乐详情页，能通过音乐详情页切换播放歌曲

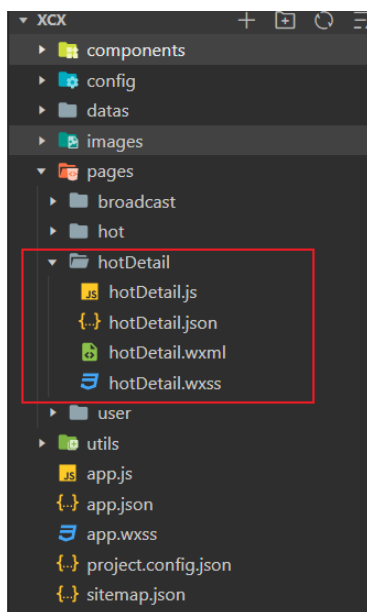
(1) 对编写音乐详情页进行分析：

- i. 音乐详情页的功能应该是点击「热歌推荐」页面的歌手歌单或者「精选电台」页面中的电台之后跳转到相应的详情页。
- ii. 音乐详情页不需要在顶部标签栏配置路由（快速入口），但是需要有返回来返回之前的页面

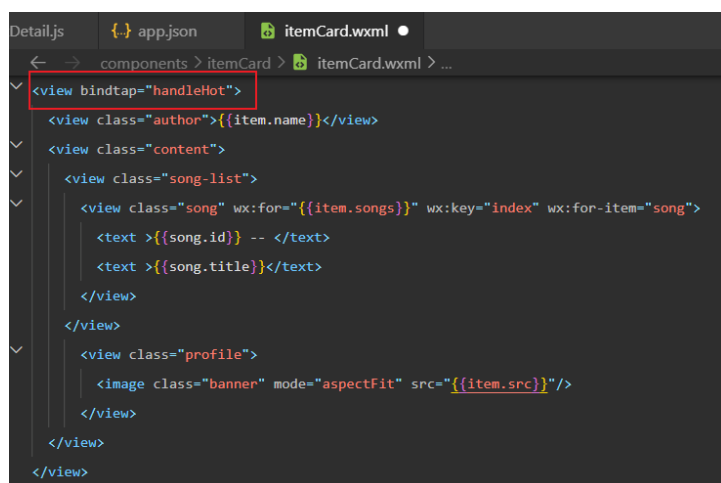
接下来以「热歌推荐」的详情页为例：

(2) hotDetail 页面设计

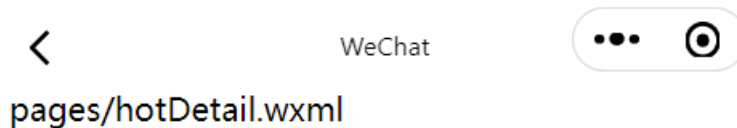
- i. 新建 hotDetail page 页面：



- ii. 为了能点击 itemCard 跳转到相应的详情页，为 itemCard 添加事件：



点击之后「热歌推荐」的卡片之后如果能显示下图效果，并能返回，说明添加成功。



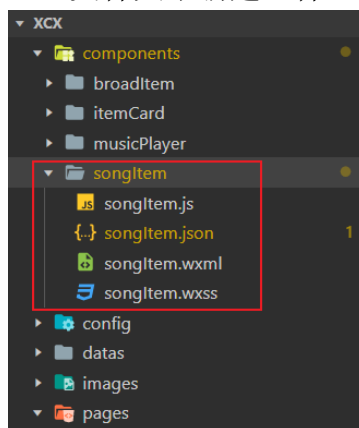
iii. 编写 hotDetail 的 wxml 文件:

通过分析得知，渲染某位歌手的歌曲信息也是在重复渲染单个歌曲组件形成一个列表，假定显示歌曲信息的组件为 songItem，它的具体信息稍后定义:

```
<!--pages/hotDetail.wxml-->
<view class='music-detail'>
  <view class='detail-banner' style='background-color: {{color}};'>
    <image src="{{poster}}" mode="aspectFill"></image>
  </view>
  <view class='detail-wrap'>
    <view wx:for="{{dataSource}}" wx:for-item="item" wx:for-index="index" wx:key="index">
      <songItem item="{{item}}" />
    </view>
  </view>
</view>
<musicPlayer />
```

iv. 编写 songItem 组件:

在 components 文件夹中新建组件 songItem,



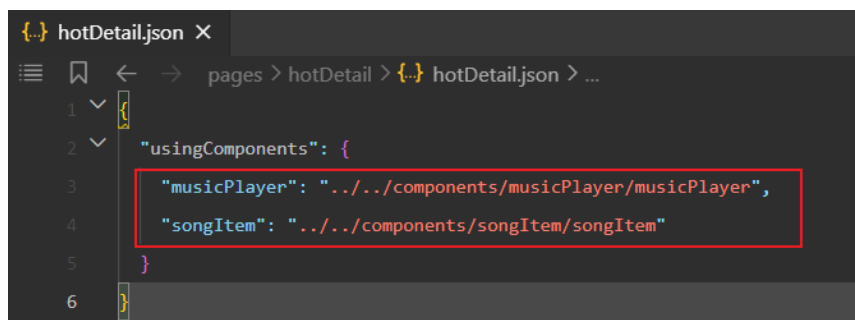
- v. 按照上一个实验步骤的方法，在 `datas` 文件夹中添加歌手歌曲信息源 `songList.js`（此处以李荣浩为例，其他歌手的请自行添加）：

```
var liRongHao = [
  {
    "id": 1,
    "name": "太坦白",
    "src": "https://music.163.com/song/media/outer/url?id=27731177.mp3",
    "poster": "https://y.gtimg.cn/music/photo_new/T002R300x300M000004Ah3HV3slJN.jpg?max_age=2592000"
  },
  {
    "id": 2,
    "name": "不将就",
    "src": "https://music.163.com/song/media/outer/url?id=31654343.mp3",
    "poster": "https://y.gtimg.cn/music/photo_new/T002R300x300M000001f11zG0EjU2u.jpg?max_age=2592000"
  }
]

var zhouJieLun = []
var xuSong = []
var songList = [
  {
    "singer": "李荣浩",
    "songs": liRongHao
  },
  {
    "singer": "周杰伦",
    "songs": zhouJieLun
  },
  {
    "singer": "许嵩",
    "songs": xuSong
  }
]

module.exports = {
  liRongHao: liRongHao,
  zhouJieLun: zhouJieLun,
  xuSong: xuSong,
  songList: songList
}
```

- vi. 在 `hotDetail` 页面中引入 `musicPlayer` 组件和 `songItem` 组件：



```
{
  "usingComponents": {
    "musicPlayer": "../../components/musicPlayer/musicPlayer",
    "songItem": "../../components/songItem/songItem"
  }
}
```

### (3) hot 页面向 hotDetail 页面传递参数

为了能从 `hot` 页面向 `hotDetail` 页面传递参数，我们在跳转的时候需要带上参数



i. 修改 itemCard.js ，在跳转时带上参数：

```
/**
 * 组件的方法列表
 */
methods: {
  handleHot: function(e) {
    wx.navigateTo({
      url: `../hotDetail/hotDetail?singer=${this.data.item.name}&poster=${this.data.item.src}`,
    })
  }
}
```

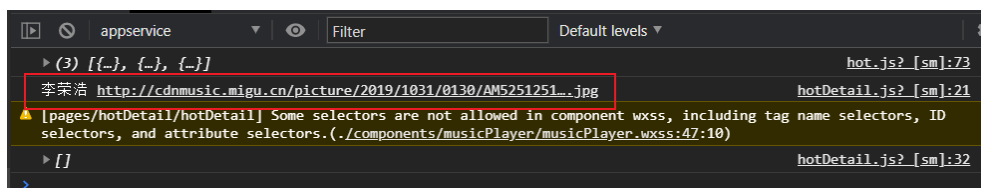
注意：两边看着像单引号的符号并不是单引号，而是反引号。

ii. 在 hotDetail.js 中接收参数：

```
data: {
  singer: null,
  poster: null,
  dataSource: [],
},

/**
 * 生命周期函数--监听页面加载
 */
onLoad: function (options) {
  const {
    singer,
    poster
  } = options;
  console.log(singer, poster);
  this.setData({
    singer: singer,
    poster: poster,
  })
},
```

iii. 如果在依次点击歌单跳转到详情页后，控制台输出如下信息，则说明传值成功。



```

(3) [{"singer": "李荣浩", "poster": "http://cdnmusic.migu.cn/picture/2019/1031/0130/AM5251251...jpg"}] hot.js? [sm]:73
李荣浩 http://cdnmusic.migu.cn/picture/2019/1031/0130/AM5251251...jpg hotDetail.js? [sm]:21
[pages/hotDetail/hotDetail] Some selectors are not allowed in component wxss, including tag name selectors, ID selectors, and attribute selectors. (./components/musicPlayer/musicPlayer.wxss:47:10)
[] hotDetail.js? [sm]:32

```

iv. 根据歌手，为详情页提供歌曲列表：

在 hotDetail.js 中引入歌曲数据：

```
// pages/hotDetail.js
import songlist from "../../datas/songList.js";
```

根据歌手为 dataSource 赋值：

```
/**
 * 生命周期函数--监听页面加载
 */
onLoad: function (options) {
  const {
    singer,
    poster
  } = options;
  console.log(singer, poster);
  this.setData({
    singer: singer,
    poster: poster,
  });

  switch (singer) {
    case "周杰伦":
      this.setData({
        dataSource: songList.zhouJieLun
      });
      break;
    case "李荣浩":
      this.setData({
        dataSource: songList.liRongHao
      });
      break;
    case "许嵩":
      this.setData({
        dataSource: songList.xuSong
      });
      break;
  }
},
```

(4) hotDetail 页面向其中的组件 songItem 传值：

i. 修改 hotDetail.wxml :

```
<!--pages/hotDetail.wxml-->
<view class='music-detail'>
  <view class='detail-banner' style='background-color: {{color}};'>
    <image src="{{poster}}" mode="aspectFill"></image>
  </view>
  <view class='detail-wrap'>
    <view wx:for="{{dataSource}}" wx:for-item="item" wx:for-index="index" wx:key="index">
      <songItem item="{{item}}" singer="{{singer}}" />
    </view>
  </view>
</view>
</musicPlayer />
```

- ii. 修改 songItem.js，在 properties 中接收 hotDetail 传值：

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

- iii. 在 songItem.js 中添加生命周期函数，查看跳转时是否获取到值（注意，在 page 中什么周期函数为 onReady，而在 component 中，生命周期函数为 ready）：

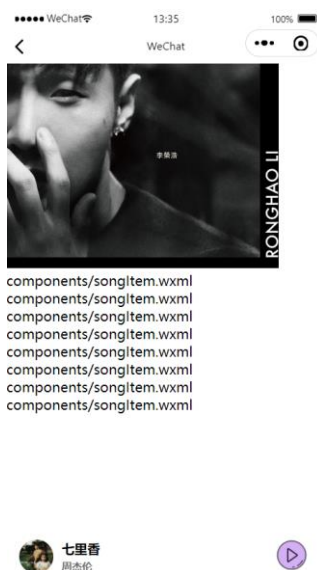
点击「热歌推荐」页面中卡片，跳转到详情页时控制台出现如

```
ready: function () {
  console.log(this.properties.item, this.properties.singer);
},
```

下信息，则说明获取值成功：

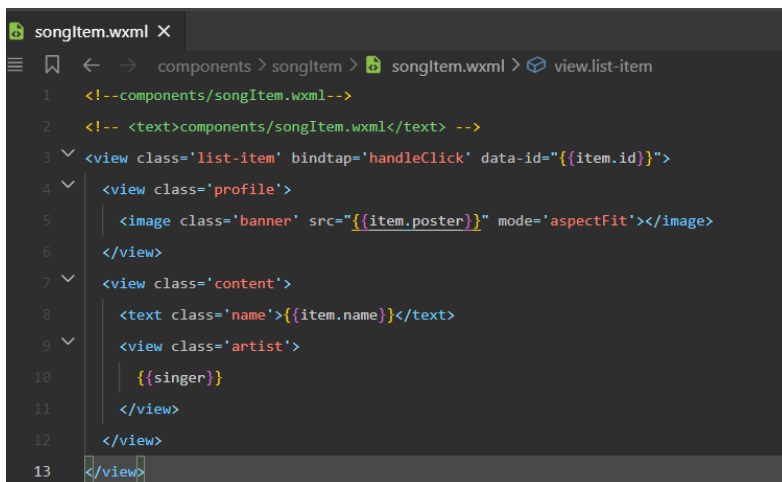


此时，详情页的效果为：



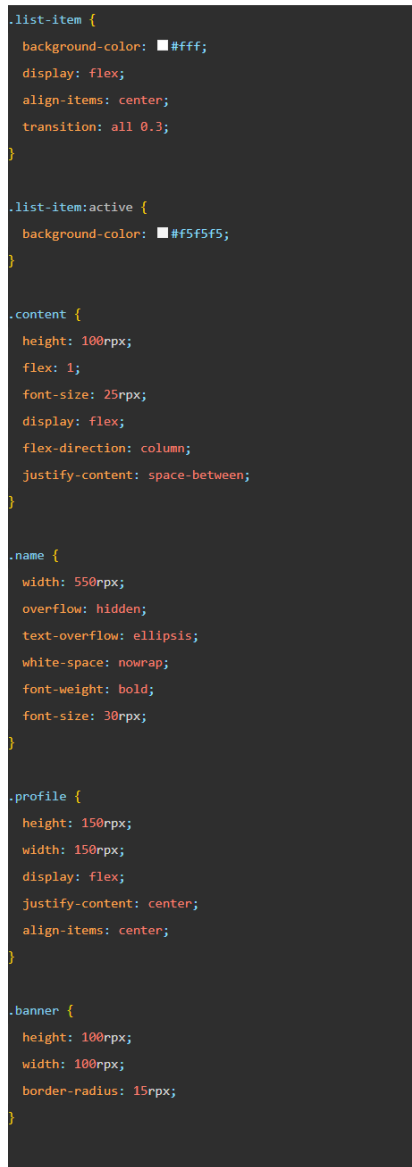
## (5) 为详情页 songItem 编写 wxml 和 wxss 文件

## i. 编写 songItem.wxml 文件:



```
1 <!--components/songItem.wxml-->
2 <!-- <text>components/songItem.wxml</text> -->
3 <view class='list-item' bindtap='handleClick' data-id='{{item.id}}'>
4   <view class='profile'>
5     <image class='banner' src='{{item.poster}}' mode='aspectFit'></image>
6   </view>
7   <view class='content'>
8     <text class='name'>{{item.name}}</text>
9     <view class='artist'>
10       {{singer}}
11     </view>
12   </view>
13 </view>
```

## ii. 编写 songItem.wxss 文件:



```
.list-item {
  background-color: #fff;
  display: flex;
  align-items: center;
  transition: all 0.3s;
}

.list-item:active {
  background-color: #f5f5f5;
}

.content {
  height: 100rpx;
  flex: 1;
  font-size: 25rpx;
  display: flex;
  flex-direction: column;
  justify-content: space-between;
}

.name {
  width: 550rpx;
  overflow: hidden;
  text-overflow: ellipsis;
  white-space: nowrap;
  font-weight: bold;
  font-size: 30rpx;
}

.profile {
  height: 150rpx;
  width: 150rpx;
  display: flex;
  justify-content: center;
  align-items: center;
}

.banner {
  height: 100rpx;
  width: 100rpx;
  border-radius: 15rpx;
}
```

完成之后的截图应如下图所示：



(6) 调整 hotDetail 页面的样式：

```
/* pages/hotDetail.wxss */
page {
  height: 100%;
}

.music-detail {
  padding-bottom: 110rpx;
}

.detail-banner {
  width: 100%;
  height: 500rpx;
  line-height: 500rpx;
  position: fixed;
  top: 0;
  z-index: -1;
}

.detail-banner image {
  width: 100%;
  height: 500rpx;
}

.detail-wrap {
  padding-top: 500rpx;
}
```

完成之后的样式应该为:



(7) 点击 songItem 使得能够切换播放歌曲:

- i. 在 hotDetail.js 中为播放器添加唯一 id, 便于获取播放器实例:

```
<!--pages/hotDetail.wxml-->
<view class='music-detail'>
  <view class='detail-banner' style='background-color: {{color}};'>
    <image src="{{poster}}" mode="aspectFill"></image>
  </view>
  <view class='detail-wrap'>
    <view wx:for="{{dataSource}}" wx:for-item="item" wx:for-index="index" wx:key="index">
      <songItem item="{{item}}" singer="{{singer}}" />
    </view>
  </view>
</view>
<musicPlayer id="player" />
```

- ii. 在 app.js 中添加全局播放器变量, 便于更新状态:

```
globalData: {
  userInfo: null,
  audio: wx.createInnerAudioContext(),
  playState: DEFAULT_MUSIC.DEFAULT_MUSIC.playState,
  musicPic: DEFAULT_MUSIC.DEFAULT_MUSIC.musicPic,
  musicName: DEFAULT_MUSIC.DEFAULT_MUSIC.musicName,
  musicUrl: DEFAULT_MUSIC.DEFAULT_MUSIC.musicUrl,
  artistName: DEFAULT_MUSIC.DEFAULT_MUSIC.artistName,
  musicPlayer: null,
}
```

iii. 在 hotDetail.js 的生命周期函数中为全局播放器变量赋初值:

```
var app = getApp();
Page({
  /**
   * 页面的初始数据
   */
  data: {
    singer: null,
    poster: null,
    dataSource: [],
  },

  /**
   * 生命周期函数--监听页面加载
   */
  onLoad: function (options) {
    var player = this.selectComponent("#player");
    console.log(player);
    app.globalData.musicPlayer = player;
  }
});
```

iv. 在 songItem.js 中编写点击切换歌曲的事件处理函数:

```
songItem.js X
components > songItem > songItem.js > properties
1 // components/songItem.js
2 var app = getApp();
3 Component({
4 /**
```

```
methods: {
  handleClick: function () {
    var musicPlayer = app.globalData.musicPlayer;
    console.log(musicPlayer);


    // 更新全局播放器变量的数据
    app.globalData.playState = 1;
    app.globalData.musicPic = this.properties.item.poster;
    app.globalData.musicName = this.properties.item.name;
    app.globalData.musicUrl = this.properties.item.src;
    app.globalData.artistName = this.properties.singer;

    // 同步当前页播放器的数据
    musicPlayer.setData({
      playState: app.globalData.playState,
      musicPic: app.globalData.musicPic,
      musicName: app.globalData.musicName,
      musicUrl: app.globalData.musicUrl,
      artistName: app.globalData.artistName
    });

    // console.log(musicPlayer.data)
    // 数据更新完毕, 切换歌曲
    musicPlayer.change();
  },
}
```

此时，播放器组件的 `change()` 方法还没有定义，接下来定义 `change` 方法：

- v. 在 `musicPlayer.js` 中定义 `change` 方法：



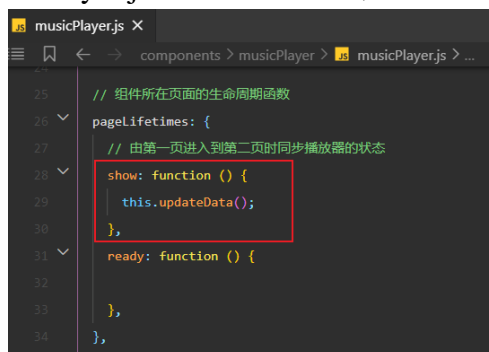
```

53 },
54 change: function () { // 切换歌曲
55   app.globalData.audio.src = app.globalData.musicUrl
56   app.globalData.playState = 1;
57   this.setData({
58     playState: 1
59   });
60   app.globalData.audio.play();
61 },
62 }
63

```

- vi. 此时在返回「热歌推荐」页面时发现播放器发生变化，不是我们当前播放的歌曲，发生这种情况的原因是我们在每一个页面都引入了一个播放器，它们是互相独立的，所以此时需要使用全局数据对播放器的数据和视图进行同步。

在 `musicPlayer.js` 的生命周期函数中添加：

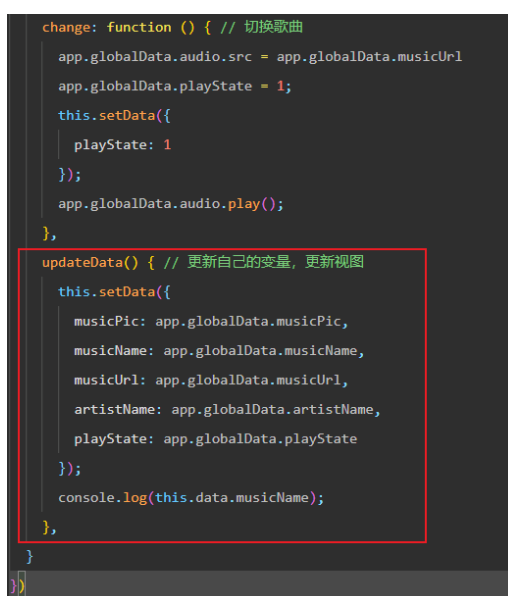


```

25 // 组件所在页面的生命周期函数
26 pageLifetimes: {
27   // 由第一页进入到第二页时同步播放器的状态
28   show: function () {
29     this.updateData();
30   },
31   ready: function () {
32
33   },
34 },

```

同时定义 `updateData` 函数：



```

change: function () { // 切换歌曲
  app.globalData.audio.src = app.globalData.musicUrl
  app.globalData.playState = 1;
  this.setData({
    playState: 1
  });
  app.globalData.audio.play();
},

updateData() { // 更新自己的变量，更新视图
  this.setData({
    musicPic: app.globalData.musicPic,
    musicName: app.globalData.musicName,
    musicUrl: app.globalData.musicUrl,
    artistName: app.globalData.artistName,
    playState: app.globalData.playState
  });
  console.log(this.data.musicName);
},
}

```

- vii. 此时，点击歌曲切换播放测试，切换页面也应该能显示正常

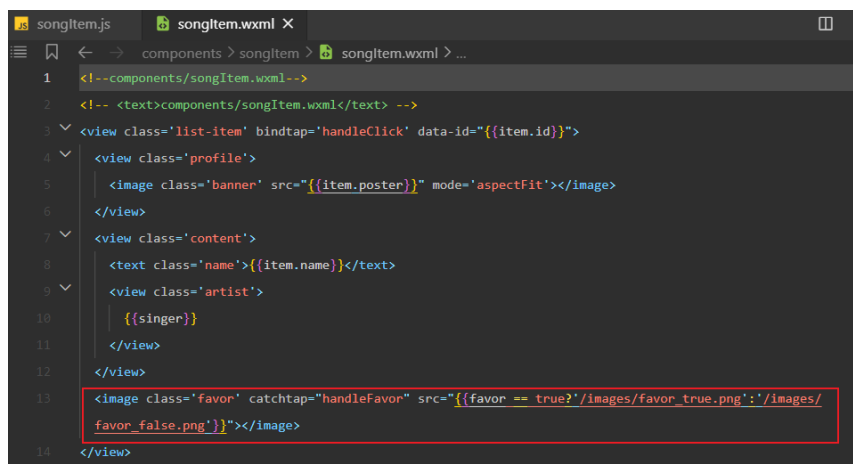


## 5. 使用小程序云开发，保存音乐收藏状态

在热歌详情页面能够对某个歌曲进行收藏，在切换页面或者重新打开小程序时，也能够显示之前已收藏歌曲的收藏状态。

(1) 在音乐详情页添加收藏按钮：

i. 修改 songItem.wxml 添加收藏图标：



ii. 在阿里巴巴图标库中选择相应的图标样式，并放置在上图路径中

iii. 修改 songItem.wxss 为收藏按钮添加样式：

```
.favor {
  width: 60rpx;
  height: 60rpx;
  padding-right: 20rpx;
}
```

iv. 为收藏操作添加数据和绑定事件：

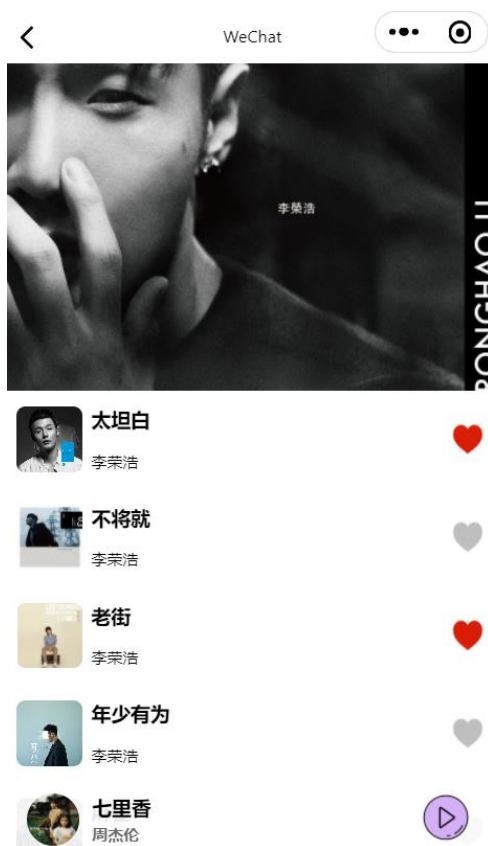
添加事件 handleFavor，并阻止事件向上冒泡：

```
<image class='favor' catchtap="handleFavor" src="{{favor == true ? '/images/favor_true.png' : '/images/favor_false.png'}}"></image>
```

```
/**
 * 组件的初始数据
 */
data: {
  favor: false,
},
```

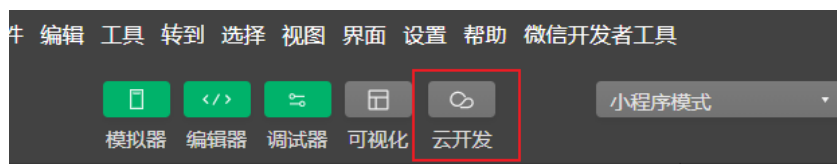
```
handleFavor: function () { // 收藏/取消收藏
  this.setData({
    favor: !this.data.favor
  })
},
```

完成效果如下图所示：



## (2) 配置小程序云开发环境

- i. 点击云开发控制台，申请云开发，并设置云开发控制台环境：



- ii. 修改项目配置，使本项目支持云开发：




在任意文件右键选择在资源管理器显示

名称	修改日期	类型	大小
components	2021/9/26 11:41	文件夹	
config	2021/9/20 13:43	文件夹	
datas	2021/9/26 11:44	文件夹	
images	2021/9/26 14:58	文件夹	
pages	2021/9/21 19:16	文件夹	
utils	2021/9/19 15:45	文件夹	
.DS_Store	2019/11/24 22:31	DS_STORE 文件	7 KB
app.js	2021/9/26 14:11	JavaScript 源文件	2 KB
app.json	2021/9/21 19:16	JSON 源文件	1 KB
app.wxss	2019/11/24 21:52	WXSS 文件	1 KB
project.config.json	2021/9/19 10:19	JSON 源文件	2 KB
sitemap.json	2019/11/24 21:52	JSON 源文件	1 KB

在顶层目录新建文件夹 miniprogram，将除了 projec.config.json 之外的所有文件和文件夹都放入 miniprogram 文件夹中：

	miniprogram	2021/9/26 15:28	文件夹
	project.config.json	2021/9/19 10:19	JSON 源文件

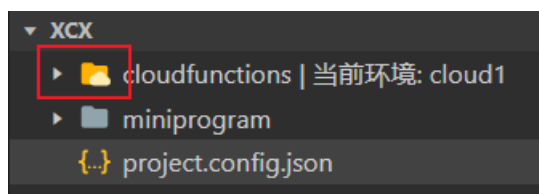
新建与 miniprogram 同级的文件夹 cloudfunctions：

	cloudfunctions	2021/9/26 15:28	文件夹
	miniprogram	2021/9/26 15:28	文件夹
	project.config.json	2021/9/19 10:19	JSON 源文件

iii. 设置云函数路径，在 project.config.json 中添加两行：



此时如果文件夹图标变化，说明配置文件生效。如果此时编译后页面无法显示，则先点击右上角的清缓存按钮，清除全部缓存，在进行编译：



(3) 初始化云环境：

在 app.js 的 onLaunch 说明周期函数中添加：

```

//app.js
import DEFAULT_MUSIC from './config/index';
App({
  onLaunch: function () {
    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 参数说明：
        // env 参数决定接下来小程序发起的云开发调用 (wx.cloud.xxx) 会默认请求到哪个云环境的资源
        // 此处请填入环境 ID，环境 ID 可打开云控制台查看
        // 如不填则使用默认环境（第一个创建的环境）
        env: 'course-demo',
        traceUser: true,
      })
    }
  }
})

// 展示本地存储能力
// var logs = wx.getStorageSync('logs') || []
// logs.unshift(Date.now())
// wx.setStorageSync('logs', logs)

```

并将开发者工具默认生成的无用函数注释掉。

(4) 在云控制台中新建集合 music\_favor:



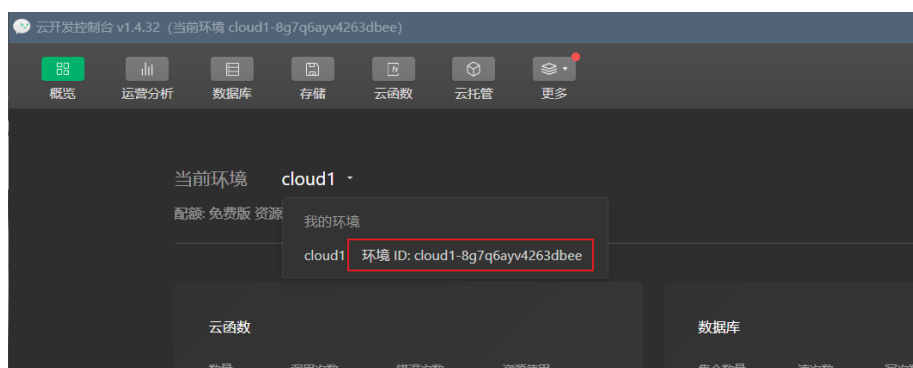
(5) 编写收藏和取消收藏插入数据库的操作:

i. 修改 songItem.js 中 handleFavor 函数:

```
handleFavor: function () { // 收藏/取消收藏
  this.setData({
    favor: !this.data.favor
  })
  console.log(this.data.favor)
  if (this.data.favor) {
    this.favorMusic();
  } else {
    this.disfavorMusic();
  }
},
}
```

ii. 编写 favorMusic 函数:

声明云对象时，需要用到云环境 ID，云函数 ID 可在云开发平台上获取，如下所示:

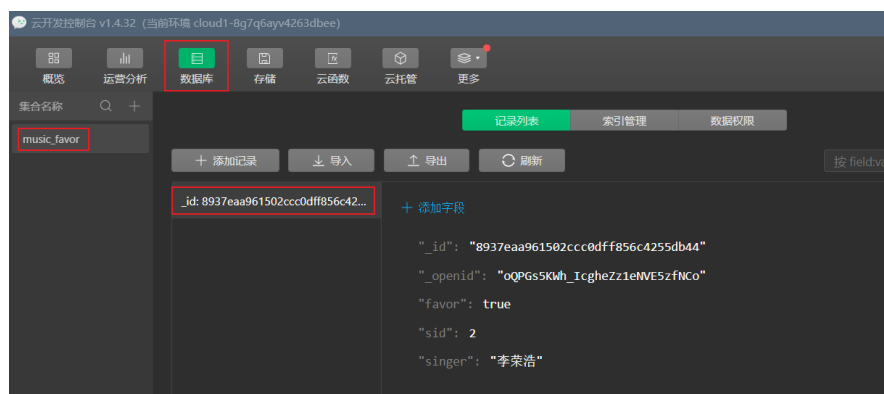


```

favorMusic: function () { // 收藏音乐, 更新全局变量并添加到云数据库
  // 添加到云数据库
  const db = wx.cloud.database({
    env: 'cloud1-8g7q6ayv4263dbee'
  });
  db.collection('music_favor').add({
    data: {
      sid: this.properties.item.id,
      singer: this.properties.singer,
      favor: true
    },
    success: res => {
      // 在返回结果中会包含新创建的记录的 _id
      this.setData({
        counterId: res._id,
        count: 1
      })
      wx.showToast({
        title: '已收藏',
      })
      console.log('[数据库] [新增记录] 成功, 记录 _id: ', res._id)
    },
    fail: err => {
      wx.showToast({
        icon: 'none',
        title: '新增记录失败'
      })
      console.error('[数据库] [新增记录] 失败: ', err)
    }
  })
},

```

点击收藏按钮, 查看模拟器和调试器输出, 并查看云数据库中是否新增了一条数据, 如下图所示:



## iii. 编写 disfavorMusic 函数:

```

disfavorMusic: function () { // 取消收藏音乐, 更新全局变量并从云数据库删除
  if (this.data.counterId) {
    // 从云数据库删除
    const db = wx.cloud.database({
      env: 'cloud1-8g7q6ayv4263dbce'
    })
    db.collection('music_favor').doc(this.data.counterId).remove({
      success: res => {
        wx.showToast({
          title: '已取消收藏',
        })
        this.setData({
          counterId: '',
          count: null,
        })
      },
      fail: err => {
        wx.showToast({
          icon: 'none',
          title: '删除失败',
        })
        console.error('[数据库] [删除记录] 失败: ', err)
      }
    })
  } else {
    wx.showToast({
      title: '无counterId, 该歌曲还未收藏',
    })
  }
}

```

点击取消收藏按钮, 查看控制台和模拟器输出, 并查看云数据库中之前的记录是否被删除。

## (6) 切换页面查看收藏的音乐能否正常显示

回到「热歌推荐」页面后再切换到热歌详情页, 原来已收藏的歌曲图标并没有显示收藏, 这是每次点开详情页每个 songItem 都是重新生成的, 并不是之前的页面, 所以我们需要将之前的状态和数据同步到新加载的页面。

## i. 添加保存收藏歌曲信息的全局变量 favorMusics:

```

globalData: {
  userInfo: null,
  audio: wx.createInnerAudioContext(),
  playState: DEFAULT_MUSIC.DEFAULT_MUSIC.playState,
  musicPic: DEFAULT_MUSIC.DEFAULT_MUSIC.musicPic,
  musicName: DEFAULT_MUSIC.DEFAULT_MUSIC.musicName,
  musicUrl: DEFAULT_MUSIC.DEFAULT_MUSIC.musicUrl,
  artistName: DEFAULT_MUSIC.DEFAULT_MUSIC.artistName,
  musicPlayer: null,
  favorMusics: {},
}

```

- ii. 利用生命周期函数，在小程序加载的时候，从数据库中获取已收藏歌曲的信息，并保存在 favorMusics 中：

在 app.js 中添加：

```
// }
// })

this.globalData.audio.src = DEFAULT_MUSIC.DEFAULT_MUSIC.musicUrl
this.getFavorMusics();
},
globalData: {
  userInfo: null,
  audio: wx.createInnerAudioContext(),
  playState: DEFAULT_MUSIC.DEFAULT_MUSIC.playState
},
getFavorMusics: function () {
  console.log("getFavorMusics")
  const db = wx.cloud.database({
    env: 'cloud1-8g7q6ayv4263dbec'
  })
  // 查询当前用户所有的 counters
  db.collection('music_favor').field({
    sid: true,
    _id: true,
  }).get({
    success: res => {
      let musics = new Array(res.data.length + 1);
      let counterIds = new Array(res.data.length + 1);

      res.data.forEach(function (item, index) {
        musics[item.sid] = true
        counterIds[item.sid] = item._id
      })
      console.log(musics, counterIds)
      this.globalData.favorMusics = {
        musics: musics,
        counterIds: counterIds
      }
      console.log(this.globalData.favorMusics);
    },
    fail: err => {
      wx.showToast({
        icon: 'none',
        title: '查询记录失败'
      })
      console.error('[数据库] [查询记录] 失败: ', err)
    }
  })
},
globalData: {
  userInfo: null,
```

- iii. 利用生命周期函数中为 songItem 组件在加载时从全局变量 favorMusics 处获取该歌曲是否已经被收藏的信息：

在 songItem.js 中修改并添加：

```
data: {
  favor: false,
},

pagelifetimes: {
  // 组件所在页面的生命周期函数
  show: function () {
    let states = app.globalData.favorMusics;
    let index = this.properties.item.id;
    this.setData({
      favor: states.musics[index],
      counterId: states.counterIds[index],
    })
  }
},
```

iv. 在 songItem.js 中修改添加收藏的函数：

此处需要注意访问数据库是异步操作，所以需要将更新全局变量的操作放在执行成功的回调函数 success 中

```
favorMusic: function () { // 收藏音乐，更新全局变量并添加到云数据库
  // 添加到云数据库
  const db = wx.cloud.database({
    env: 'cloud1-8g7q6ayv4263dbee'
  });
  db.collection('music_favor').add({
    data: {
      sid: this.properties.item.id,
      singer: this.properties.singer,
      favor: true
    },
    success: res => {
      // 在返回结果中会包含新创建的记录的 _id
      this.setData({
        counterId: res._id,
        count: 1
      })
      wx.showToast({
        title: '已收藏',
      })
      console.log('[数据库] [新增记录] 成功, 记录 _id: ', res._id)
      // 更新全局变量
      app.globalData.favorMusics.musics[this.properties.item.id] = true
      app.globalData.favorMusics.counterIds[this.properties.item.id] = this.data.counterId
      // console.log("globalData", app.globalData.favorMusics)
    },
    fail: err => {
```

v. 在 songItem.js 中修改取消收藏的函数：



```

disfavorMusic: function () { // 取消收藏音乐, 更新全局变量并从云数据库删除
  if (this.data.counterId) {
    // 从云数据库删除
    const db = wx.cloud.database({
      env: 'cloud1-8g7q6ayv4263dbce'
    })
    db.collection('music_favor').doc(this.data.counterId).remove({
      success: res => {
        wx.showToast({
          title: '已取消收藏',
        })
        this.setData({
          counterId: '',
          count: null,
        })

        // 更新全局变量
        app.globalData.favorMusics.musics[this.properties.item.id] = false
        app.globalData.favorMusics.counterIds[this.properties.item.id] = null
      },
      fail: err => {
        wx.showToast({

```

(7) 至此, 重新打开小程序或者切换页面, 也能够看到之前被收藏的音乐能正确显示已被收藏的状态。

**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)

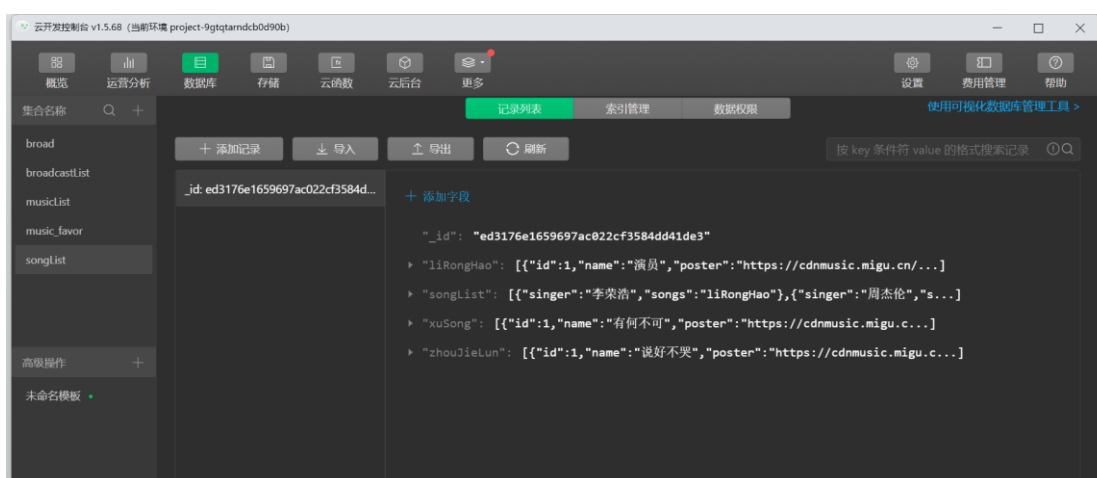
问题: 1. 被各组件、page、component 之间的数据传输绕晕了, 花了很久才理顺

2. 获取数据库里的数据也给我带来的很多困难，通过搜集各种资料，终于找到了不用重写很多代码就能把数据存在数据库里的方法

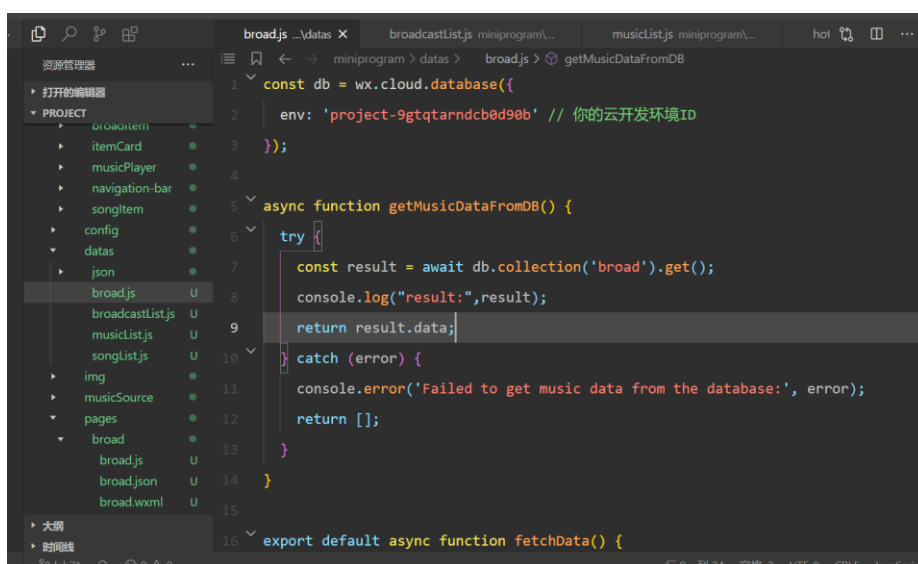
## （二）实验记录（实验过程中关键步骤截图记录及文字描述）

2.1 完成所有「实验步骤」内容，将报告中的功能和代码添加到经过上次实验所完成一半的小程序中。将之前写在配置文件中的歌曲数据插入到数据库中，并将数据引入方式改为从数据库中引入。将相关代码片段及小程序关键页面在下方进行截图展示。

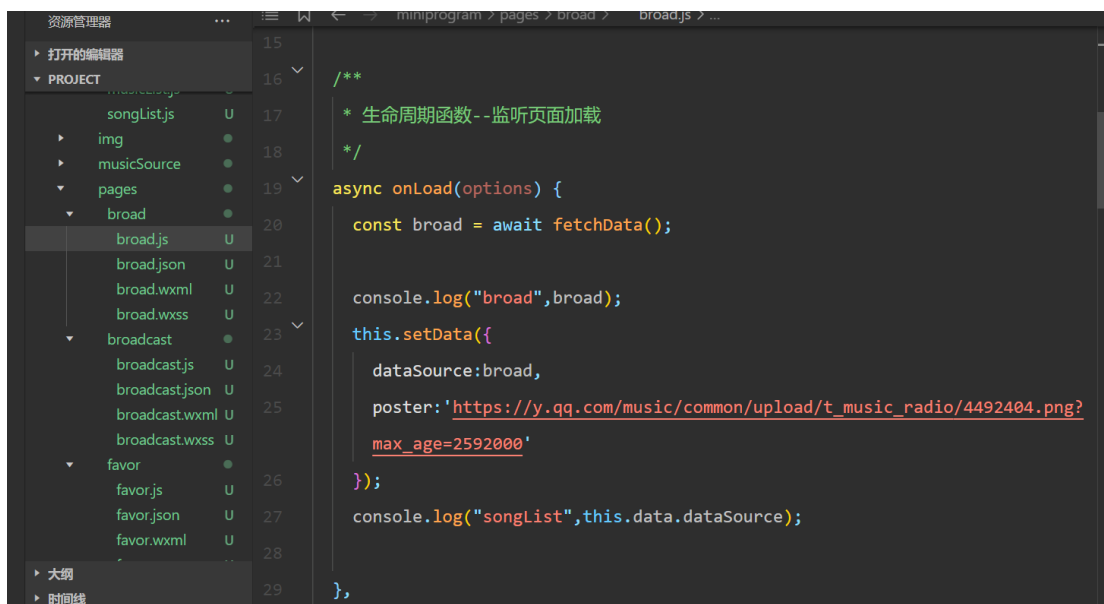
数据库里的数据：



从数据库中导入数据到本地变量：



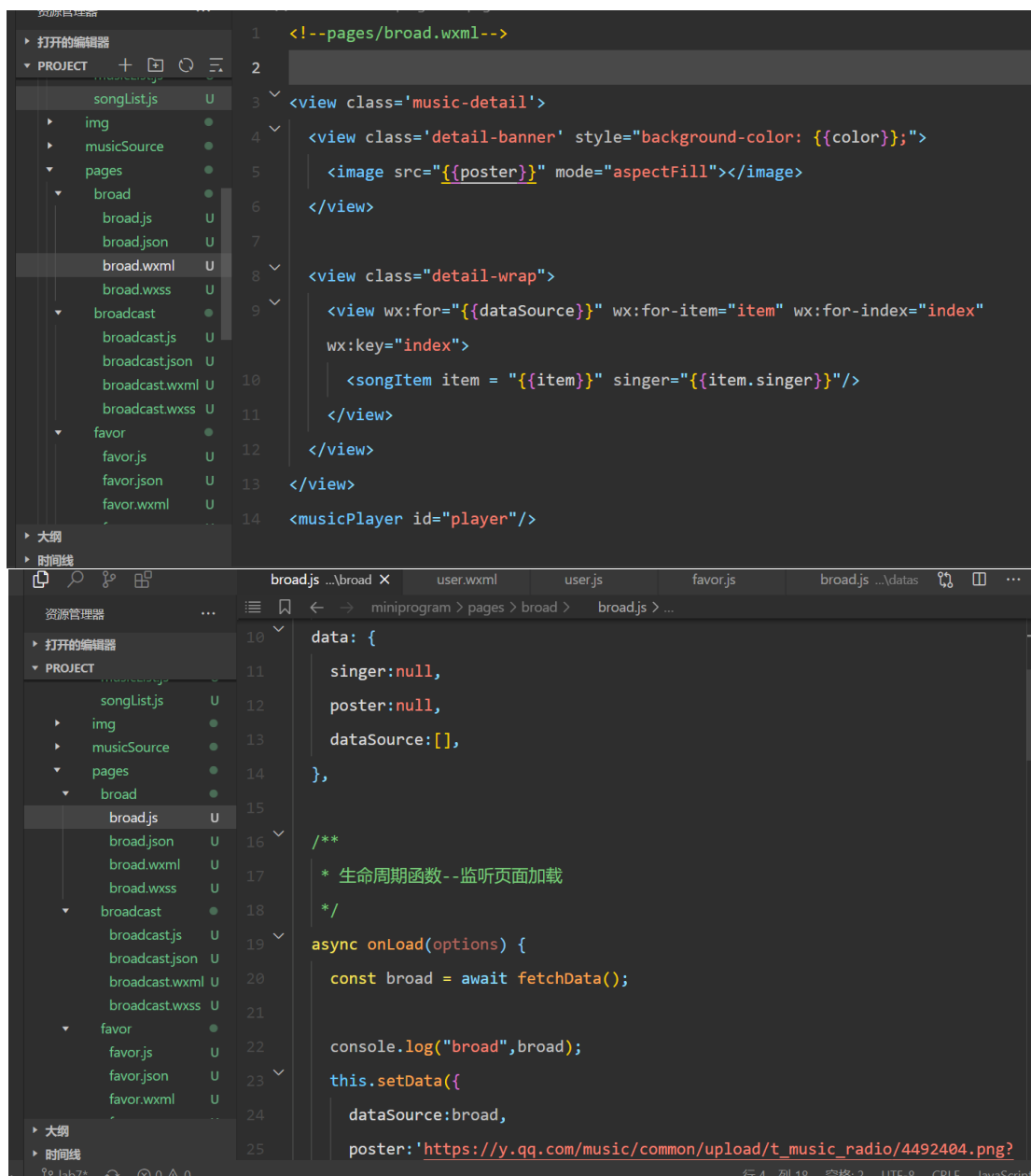
```
  const broad = await getMusicDataFromDB();
  return broad;
}
```



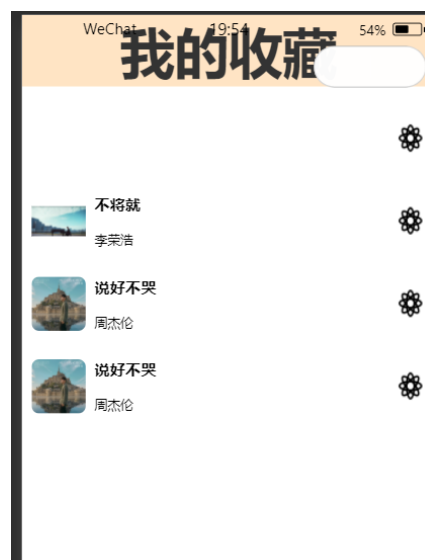
2.2 实验报告中只对「热歌推荐」页面制作了详情页，请制作「精选电台」页面歌曲的详情页，并使歌曲有收藏功能（可复用报告中提供的 `songItem` 组件）。将相关代码片段及小程序关键页面在下方进行截图展示。

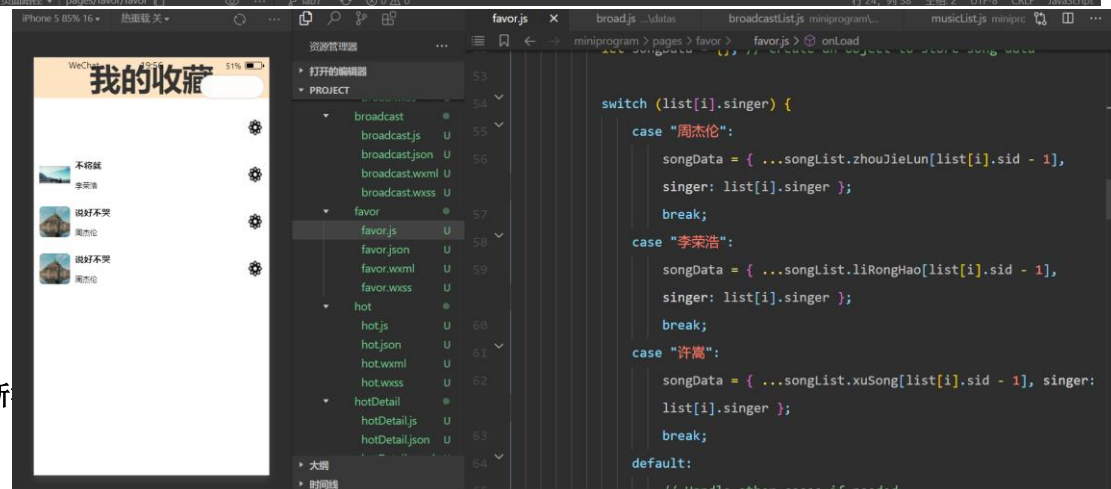
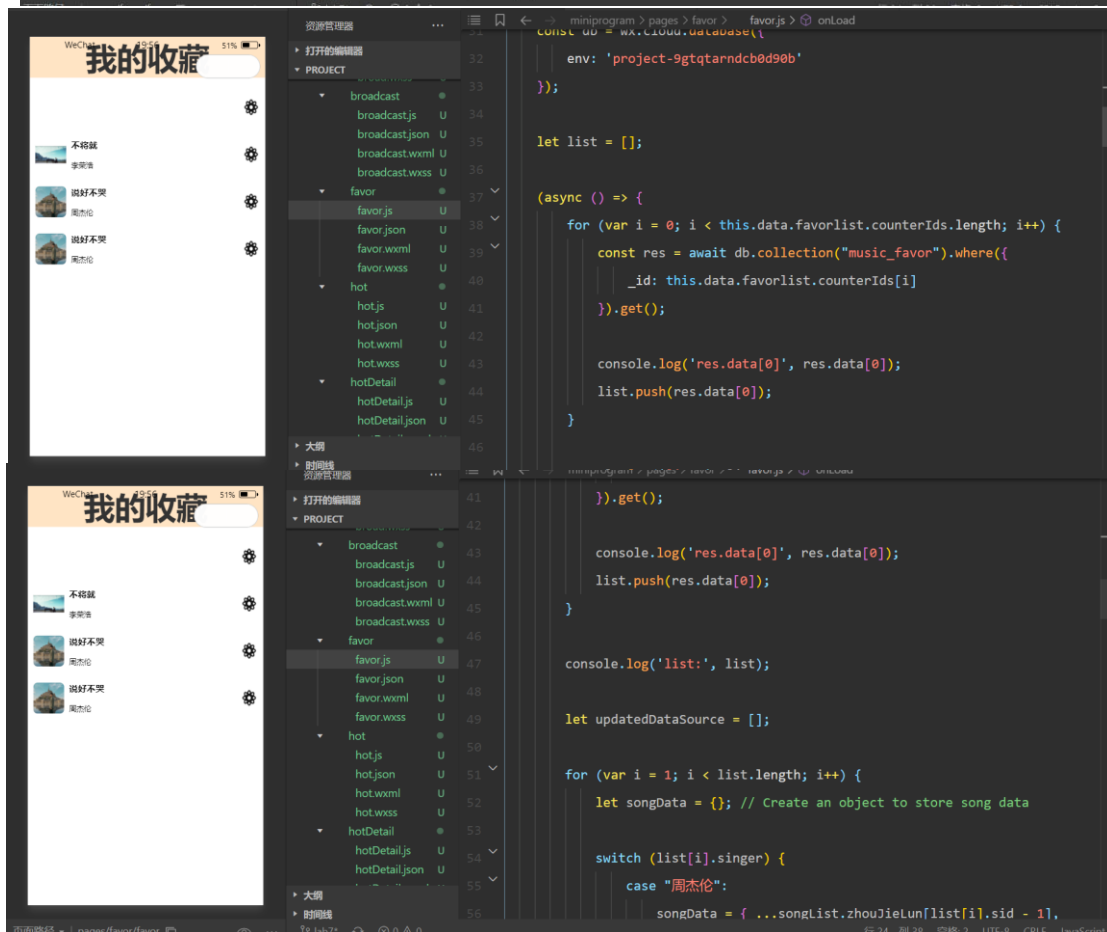
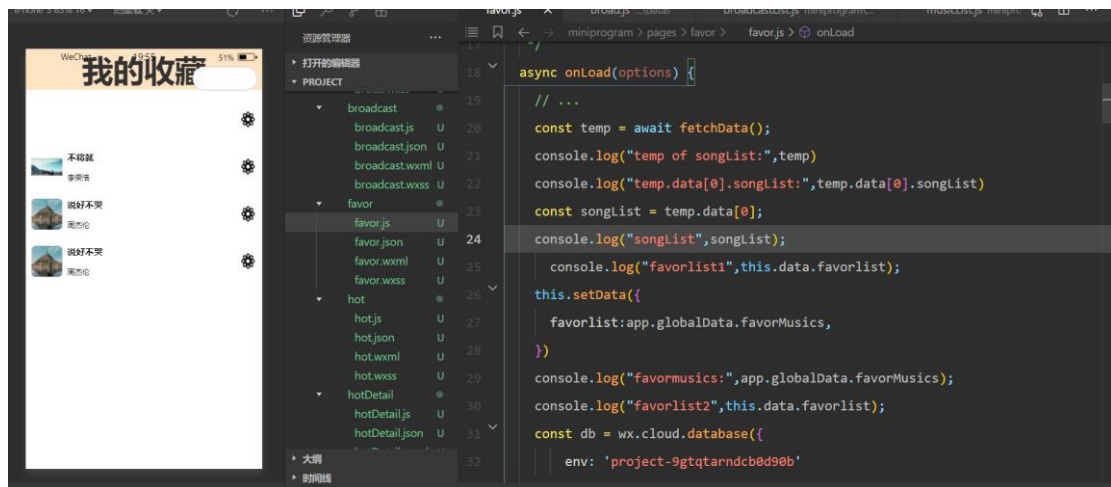
(个性电台)

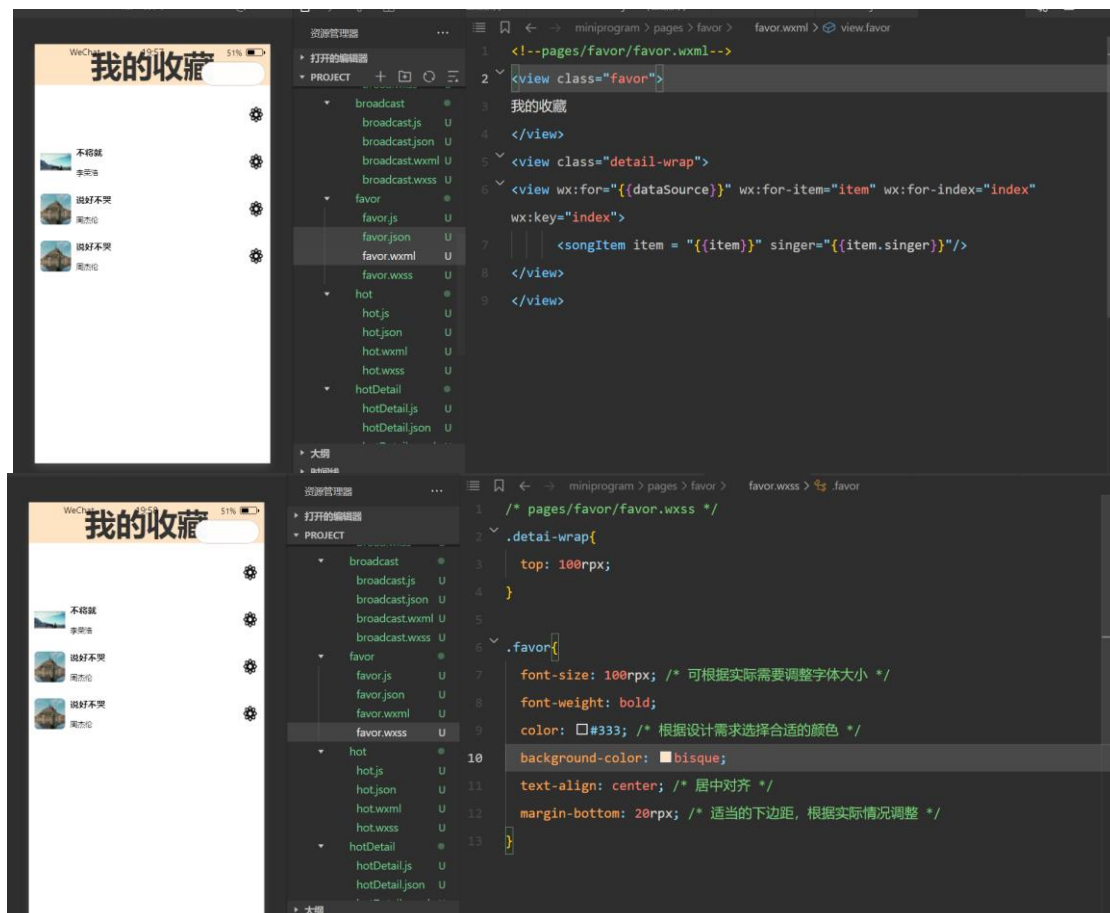




2.3 在「个人中心」页面提供查看已收藏歌曲列表的功能。将相关代码片段及小程序关键页面在下方进行截图展示。







## 2.4 上传最终的小程序源代码文件。