
灵魂音乐小程序

项目开发文档

作者：灵魂学者（LHM）

目录

一、开发环境	3
1. 微信开发者工具：下载地址	3
2. 安装 Express 框架，用于快速搭建 HTTP 服务器	3
3. 安装 nodemon 监控文件修改	3
5. 在服务器目录下启动	4
二、页面预览	5
三、功能模块说明	8
(1) 顶部模块切换（音乐、助眠、听书）	8
(2) 搜索功能模块	9
跳转的搜索页面(新歌板块)	10
搜索结果	11
历史记录	15
(3) 轮播图板块	15
(4) Top 板块	17
(5) 推荐模块	19
(6) MV 模块	21
(7) 底部播放器模块	22
(8) 播放列表	26
(9) 专辑播放页面	28
四、总结	29

一、开发环境

1.微信开发者工具：[下载地址](#)

2.Node.js 做一个轻量级服务器，如果不想用 Node.js 做服务器也可以，或者将其写成静态数据也可以，另外想做上线的也可以，需要拥有服务器，当然我们在做本地服务器的时候会出现在测试的时候是数据无法做到渲染，因为有一个“校验域名，HTTPS 协议”之类的限制，所以想要有类似的体验可以用这个内网穿透的东西，但如果还是想上线，还是需要服务器，以及做域名备案之类的工作。

Node.js :[下载地址](#) ; Git bash :[下载地址](#)



```
MINGW64:/c/Users/Administrator/Desktop
Administrator@DESKTOP-3D7435R MINGW64 ~/Desktop
$ node -v
v16.15.0
Administrator@DESKTOP-3D7435R MINGW64 ~/Desktop
$ |
```

Node 服务器搭建（简）：[服务器搭建参考](#)

1. 创建一个目录 server，使用 Git Bash (或 CMD) ,初始化项目

```
npm init -y
```

2. 安装 Express 框架，用于快速搭建 HTTP 服务器

```
npm install express --save
```

3. 安装 nodemon 监控文件修改

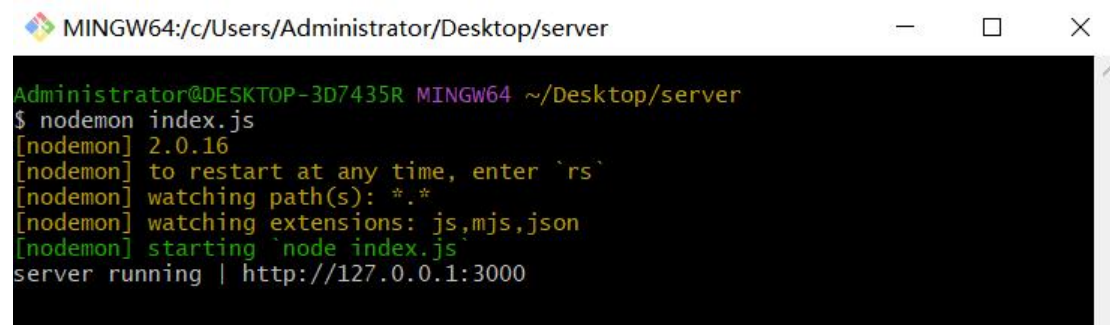
```
npm install nodemon -g
```

4. 执行完上述命令，在目录创建一个 index.js，编写请求服务

器代码。

5. 在服务器目录下启动

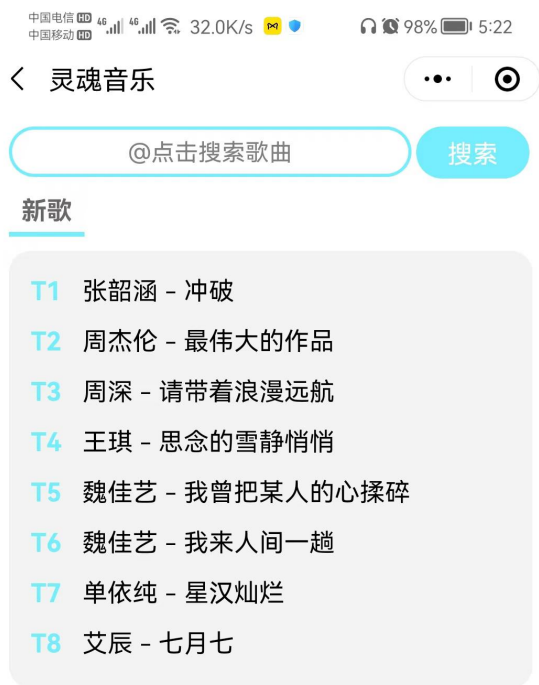
nodemon index.js

A screenshot of a terminal window titled 'MINGW64:/c/Users/Administrator/Desktop/server'. The terminal shows the command 'nodemon index.js' being executed. The output includes the version '2.0.16', instructions to restart with 'rs', the paths being watched ('*.js'), the extensions being watched ('.js, .mjs, .json'), and the message 'starting node index.js'. Finally, it displays 'server running | http://127.0.0.1:3000'.

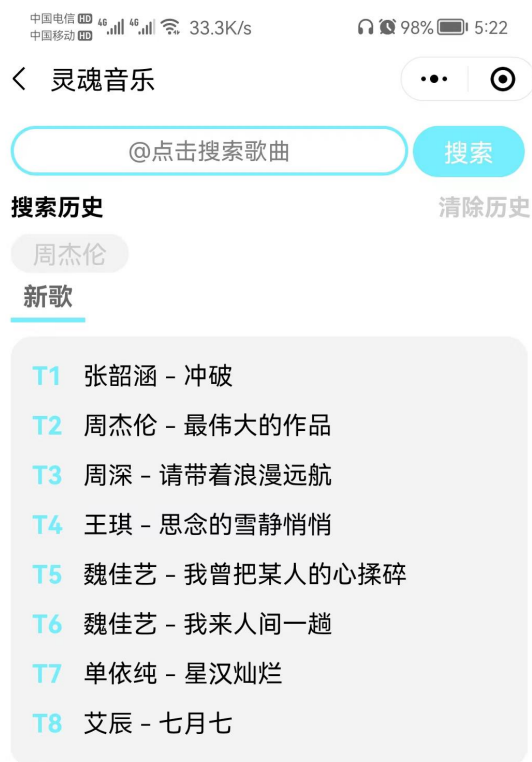
```
MINGW64:/c/Users/Administrator/Desktop/server
Administrator@DESKTOP-3D7435R MINGW64 ~/Desktop/server
$ nodemon index.js
[nodemon] 2.0.16
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node index.js`
server running | http://127.0.0.1:3000
```

以上效果则启动成功！

二、页面预览



（首页）



（搜索界面）



（搜索历史界面）

（搜索结果）



（播放列表）

（专辑页面）

三、功能模块说明

(1)顶部模块切换（音乐、助眠、听书）

音乐 助眠 听书

index.wxml 实现顶部 Tap 页面切换模块

分别对应每一个 swiper,通过每个 swiper 可以进行切换,通过 data-*: data-item 来定义页面的 item 值,可以对顶部的按钮进行点击切换,用 bindtap="changeItem"来获取点击当前的 item 值并进行切换,同时也设置了在 swiper 滑动切换的时候用 bindchange="changeTab"获取一个值,通过 current = " {{item}} "进行页面切换。

Index.wxml

```
<view class="content">
  <swiper class="content-wh" current="{{item}}" bindchange="
changeTab">
    <swiper-item> ...

<!-- 顶部区域 -->
<view class="tab">
  <view class="tab-item {{tab==0?'active':''}}" bindtap="cha
ngeItem" data-item="0">音乐</view>
  <view class="tab-item {{tab==1?'active':''}}" bindtap="cha
ngeItem" data-item="1">助眠</view>
  <view class="tab-item {{tab==2?'active':''}}" bindtap="cha
ngeItem" data-item="2">听书</view>
```

```
</view>
index.js
//定义属性，e 获取

data:{ item:0 , tap:0 }
//点击顶部(音乐/助眠/听书)时获取-响应
changeItem:function(e){
  // console.log(e);
  this.setData({
    item : e.target.dataset.item
  })
},
//滑动 Tap 时获取-响应
changeTab:function(e){
  // console.log(e);
  this.setData({
    tab : e.detail.current
  })
  // console.log(this.data.tab)
},
```

(2)搜索功能模块

这里搜索框的主要为了点击跳转到我们的搜索界面，并不是为了在此进行搜索，所以做一个点击事件跳转即可；(在 app.json 写上一个页面路径 /pages/search/search, 在这个 search.wxml 搜索页面)



```
index.wxml

<swiper-item>
  <view class="search" bindtap="toSearch">
    <input placeholder="点击搜索歌曲"/>
  </view>
</swiper-item>
```

```
</view> ...
```

```
index.js
```

```
//点击跳转
```

```
toSearch(){
  wx.navigateTo({
    url: '/pages/search/search',
    success: (res) => {
      // console.log(跳转成功)
    }
  })
},
```

跳转的搜索页面(新歌板块)



```
<view class="new clear">
  <view class="new_title"><text>新歌</text></view>
  <view class="new_content">
    <view class="new_li">
      <view class="new_name" wx:for="{{newList}}" wx:key="
id">
        <text class="tip">T{{index+1}}</text>
        <text class="ma_l"> {{item.filename}}</text>
      </view>
```

```
    </view>
  </view>
</view>
```

新歌模块是通过酷狗的 api 进行渲染展示的，在跳转页面加载时进行请求，并取前 8 条数据进行渲染展示。

```
onLoad: function (options) {
  wx.request({
    url: 'http://m.kugou.com/?json=true',
    success: (res) => {
      // console.log(res);
      this.setData({
        newList : res.data.data.slice(0,8)
      })
      // console.log(this.data.newList)
    }
  })
},
```

搜索结果



点击搜索后进行跳转到搜索结果界面，展示搜索的结

果，这里也是用酷狗的 api，通过关键词搜索，将数据进行保存接着进行页面的渲染。

```
<!-- 搜索结果 -->
<view class="content clear">
  <view class="title"><text>已搜单曲</text></view>
  <view class="ul">
    <view class="li" bindtap="listen" data-index="{{index}}"
      wx:for="{{searchList}}" wx:key="this">
      <view class="songname"><text>{{item.songname}}</text><
/ view>
      <image class="collect" src="/pages/images/search/collect.png"></image>
      <image class="mv" src="/pages/images/search/mv.png"></
image>
      <image class="more" src="/pages/images/search/more.png
"></image>
      <view class="filename"><text>{{item.filename}}</text><
/ view>
    </view>
  </view>
  <view class="foot">-- 到底了 --</view>
  <play></play>
</view>
```

search.js

点击搜索时对搜索框进行判断是否输入了内容，没有的话进行提示，有的话进行跳转，同时将搜索的关键词放到历史搜索处。

```
//点击搜索
search() {
  //判断输入框中是否为空,空则提示,不是则请求
  if (this.data.keyword != '') {
```

```

        wx.request({
          url: 'http://mobilecdn.kugou.com/api/v3/search/song?
format=json&keyword=' + this.data.keyword,
          method: 'GET',
          success: (res) => {
            // console.log(res);

            // 记录搜索

            // 记录 6 条记录,如果大于 6 条记录将大于 6 条的旧搜索删除
            if(this.data.historyList.length < 7){
              this.data.historyList.unshift(this.data.keyword)
;
            }else{
              this.data.historyList.pop(this.data.keyword)
            }

            this.setData({
              searchList: res.data.data.info,
              click: true,
              historyList : this.data.historyList
            })
            // console.log(this.data.searchList);
            // console.log(this.data.click);
            // console.log(this.data.historyList);
          }
        })
      } else {
        //空则提示
        wx.showToast({
          icon: 'error',
          title: '请输入关键字',
        })
      }
    },
  },

```

点击搜索结果，用 `data-* = data-index` 来获取点击的对象，获取该歌曲的 `hash` 值，通过 `hash` 值来发送请求获取该歌曲

所对的属性，如 id,singer,filename,src,coverImgurl ...

```
listen(e){
  console.log('listen')
  // console.log(e)
  // console.log(this.data.searchList[e.currentTarget.dataset.
  index].hash)
  this.setData({
    hash:this.data.searchList[e.currentTarget.dataset.index].h
    ash,
  })
  wx.request({
    url: 'http://m.kugou.com/app/i/getSongInfo.php?cmd=playInf
    o&hash='+this.data.hash,
    success:(res)=>{
      console.log(res)
      // console.log(res.data.backup_url[0])
      // console.log(res.data.backup_url);
      // console.log(this.data.play.coverImgUrl)
      this.data.play.coverImgUrl = res.data.album_img
      this.data.play.coverImgUrl = this.data.play.coverImgUrl.
      replace('{size}/','')
      console.log(this.data.play.coverImgUrl)
      this.setData({
        'play.id' : JSON.parse(res.data.audio_id),
        'play.singer' : res.data.author_name,
        'play.filename' : res.data.fileName,
        'play.src' : res.data.backup_url[0],
        'play.coverImgurl': res.data.album_img
      })
      // console.log(this.data.singer+'|'+this.data.filename)
      // console.log(this.data.play.id)
      wx.navigateTo({
        // url: '/pages/index/index?play='+this.data.p
        url: '/pages/index/index?id='+this.data.play.id+'&file
        name='+this.data.play.filename+'&singer='+this.data.play.sin
        ger+'&src='+this.data.play.src+'&coverImgUrl='+this.data.pla
        y.coverImgUrl,

        // 测试:  title=傅梦彤 - 潮汐 (Natural)&singer=傅梦彤
        &src=https://sharefs.tx.kugou.com/202207091055/75db08415b943
        5e7657aa57889b6abba/KGTx/CLTX001/72d3f4bdf8379f60c7563900d3e
        0080e.mp3
      })
    }
  })
}
```

```
}  
})  
}
```

历史记录

如果点击搜索则有历史记录，历史记录只记录六条，如果超过六条记录，则在第七条记录往前数，原来的第一条记录则会被清除，同时也有清除历史功能。



(3)轮播图板块

这里可以用一些 api 来做这个 banner,这里我们用以一个酷狗的 api 中的数据来做一个 banner 模块，通过拿取数据来对页面进行渲染。



API : <http://m.kugou.com/?json=true>

可通过 postman 测试查看对应的 json 数据

<!-- banner 区域 -->

```
<view class="banner">
  <swiper autoplay>
    <swiper-item wx:for="{{bannerList}}" wx:key="id">
      <image mode="widthFix" src="{{item.imgurl}}"></image>
    </swiper-item>
  </swiper>
</view>
```

index.js

在 data 当中定义一个 bannerList 空数组来接收，当然不想做这个 api 请求，也可以将数组放到我们的 bannerList 直接做。

```
data:{ bannerList:[] },
```

//轮播图获取

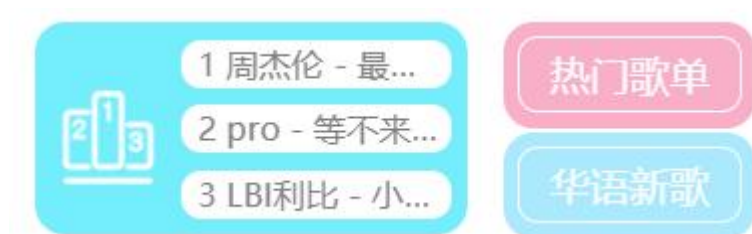
```
getBanner(){
  wx.request({
    url: 'http://m.kugou.com/?json=true',
    method: 'GET',
    success:(res)=>{
      // console.log(res);
      this.setData({
        bannerList : res.data.banner
      })
      // console.log(this.data.bannerList);
    }
  })
}
```

```
},
```

//写完这个方法，我们需要调用，在页面加载的时候（onLoad）进行调用，进行页面的一个加载，渲染轮播图。

```
onLoad: function (options) {  
    // 获取 banner 图  
    this.getBanner();  
}
```

(4)Top 板块



Top,通过浮动的方式布局，Top 这里的数据同样采用酷狗音乐的 api 数组进行渲染。

api :

<http://m.kugou.com/rank/info/?rankid=8888&page=1&json=truerl>

```
<!-- Top -->  
<view class="Top">  
    <view class="top-1">  
        <image class="fl" src="/pages/images/index/top.png"></i  
image>  
        <view class="t_songname" wx:for="{{topList}}" wx:key="i  
d">  
            <text class="filename">{{index+1}}  {{item.filename}}</  
text>  
        </view>  
    </view>  
  
    <view class="top-2"><text class="hotLi">热门歌单  
</text></view>
```

```
<view class="top-3"><text class="newSong">华语新歌
</text></view>
<view class="clear"></view>
</view>
```

```
index.js
```

```
data:{ topList:[] }
```

```
//Top 获取
```

```
getTop(){
  wx.request({
    url: 'http://m.kugou.com/rank/info/?rankid=8888&page=1&js
son=truerl',
    method: 'GET',
    success:(res)=>{
      // console.log(res);
      // console.log(res.data.songs.list);
      this.setData({
        topList : res.data.songs.list.splice(0,3)
      })
      // console.log(this.data.topList);
    }
  })
},
```

(5) 推荐模块



弹性布局，仅显示数据的前六条，保存到数组并进行渲染，（未写->点击更多跳转到新的页面，并展示数据每个推荐的歌单，点击歌单可以展示歌单里的歌曲。）

```
<!-- 推荐 -->
<view class="recommend">
  <view class="re_title"><text>推荐</text><text>更多</text></view>
  <view class="re_content">
    <view class="re_li" wx:for="{{recommendList}}" wx:key="id">
      <image src="{{item.imgurl}}"></image>
      <text class="re_filename">{{item.specialname}}</text>
      <text class="re_author">{{item.play_count_text}}</text>
    </view>
  </view>
</view>
```

```
</view>
```

```
index.js
```

可以将这里的 127.0.0.1:3000 用内网穿透的方式，换成内网穿透的网址，

在不是局域网的时候测试可以渲染数据，但上线是不可以的。

splice() 截取前 6 条数据进行渲染即可

```
data:{ recommendList:[] }
```

```
//推荐获取
```

```
getRecommend(){
  wx.request({
    url:'http://127.0.0.1:3000/recommend',
    method:'GET',
    success:(res)=>{
      // console.log(res);
      res.recommendList
      this.setData({
        // recommendList : res.data.plist.list.info.splice
(0,6)
        recommendList : res.data.data.list.splice(0,6),
      })
      // console.log(this.data.recommendList);
      // console.log(this.data.recommendList[0].imgurl);
    }
  })
},
```

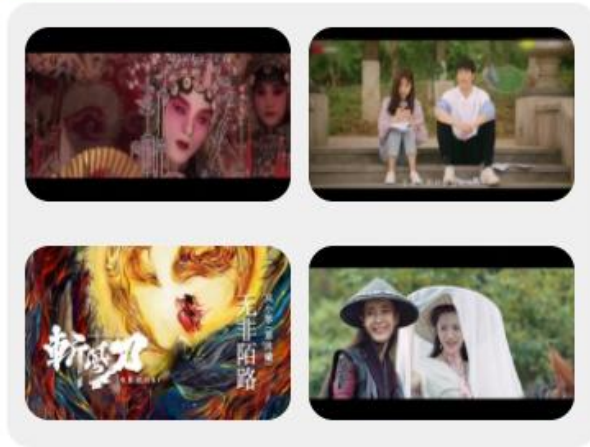
同样在 onLoad（页面加载的时候）调用，对 Top 板块进行渲染；

```
onLoad: function (options) {
  // 获取 banner 图
  this.getBanner();
  //获取 Top 歌曲
  this.getTop();
}
```

(6)MV 模块

MV视频

更多>



```
<!-- MV -->
<view class="mv">
  <view class="m_title"><text>MV 视频</text><text>更
多</text></view>
  <view class="mv_content">
    <view class="m_video" wx:for="{{mvList}}" wx:key="
id">
      <image src="{{item.imgurl}}"></image>
    </view>
  </view>
</view>
```

index.js

```
data:{ mvList:[] }
```

//MV 获取

```
getMv(){
  wx.request({
    url: 'http://mobilecdnbj.kugou.com/api/v3/singer/mv?si
ngername=%E9%A3%8E%E5%B0%8F%E7%AD%9D&pagesize=20&singerid=86
747',
    method: 'GET',
```

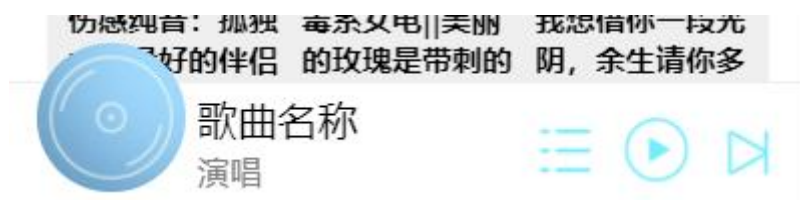
```

    success:(res)=>{
      // console.log(res);
      this.setData({
        mvList : res.data.data.info.splice(0,4)
      })
      // console.log(this.data.mvList);
    }
  })
},

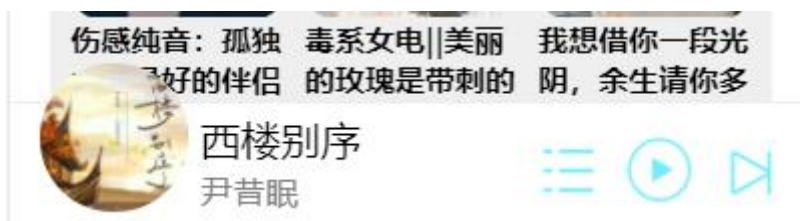
```

(7)底部播放器模块

无播放歌曲时样式:



有播放歌曲时样式:



<!-- 底部菜单 -->

```

<view class="play-menu">
  <image wx:if="{{play.coverImgUrl!=''}}" class="pic {{state=='running'? 'rotate':''}}" bindtap="exit" src="{{play.coverImgUrl}}"></image>
  <image wx:else class="pic {{state=='running'? 'rotate':''}}" bindtap="exit" src="http://img.kugou.com/stdmusic/{size}/20220104/20220104141256795835.jpg"></image>
  <view class="songname">{{play.title}}</view>
  <view class="singer">{{play.singer}}</view>
  <image class="play-list" bindtap="pList" src="/pages/images/index/list.png"></image>
  <image wx:if="{{state=='paused'}}" class="play-play" bindtap="pPlay" src="/pages/images/index/play.png"></image>

```

```
<image wx:else class="play-play" bindtap="pPause" src="/
pages/images/index/paused.png"></image>
<image class="play-next" bindtap="pNext" src="/pages/ima
ges/index/next.png"></image>
</view>
```

index.js

```
data:{
  state:'paused', //状态
  play:{
    title:'歌曲名称',

    singer:'演唱',
    coverImgUrl:'',
    currentTime:'00:00',
    duration:'00:00',
    percent:0
  },
  musicList:[]
}
```

存放音乐播放的位置 musicList:[],播放列表中获取 getMusic

```
//歌曲获取
getMusic:function(){
  wx.request({
    url:'http://23.224.99.235:16961/musicList',
    method:'GET',
    success:(res)=>{
      // console.log(res);
      this.setData({
        // musicList : res.data.data.info
      })
      console.log(this.data.musicList);
    }
  })
},
```

在 onLoad(页面加载)时调用,渲染到播放列表中去。

```
onLoad: function (options) {
  // 获取 banner 图
```

```
this.getBanner();  
//获取 Top 歌曲  
this.getTop();  
//获取推荐板块  
this.getRecommend();  
// 获取 Mv 板块  
this.getMv();  
// 获取音乐  
this.getMusic();  
console.log(options);  
this.data.musicList.push(options)  
this.setData({  
  musicList : this.data.musicList,  
})
```

音频，媒体组件 `audio` 已经停止维护，使用 `wx.createInnerAudioContext` 代替
查看文档：<https://developers.weixin.qq.com/miniprogram/dev/component/audio.html>

在页面的初次渲染完成

```
audioCtx: null,  
onReady: function () {  
  this.audioCtx = wx.createInnerAudioContext()  
  // 默认选择第 1 曲  
  this.setMusic(0);  
  var that = this;  
  // 播放进度检测  
  this.audioCtx.onError(function() {  
    console.log('播放失败: ' + that.audioCtx.src)  
  })  
  // 播放完成自动换下一曲  
  this.audioCtx.onEnded(function() {  
    that.next()  
  })
```

```
// 自动更新播放进度

this.audioCtx.onPlay(function() {})  
this.audioCtx.onTimeUpdate(function() {  
    that.setData({  
        'play.duration': formatTime(that.audioCtx.duration),  
        'play.currentTime': formatTime(that.audioCtx.current  
Time),  
        'play.percent': that.audioCtx.currentTime / that.aud  
ioCtx.duration * 100  
    })  
})  
  
// 格式化时间  
  
function formatTime(time) {  
    var minute = Math.floor(time / 60) % 60;  
    var second = Math.floor(time) % 60  
    return (minute < 10 ? '0' + minute : minute) + ':' + (  
second < 10 ? '0' + second : second)  
}  
  
},  
  
// setMusic  
setMusic:function(index){  
    var music = this.data.musicList[index]  
    // console.log(this.data.musicList[0])  
    this.audioCtx.src = music.src  
    this.setData({  
        playIndex:index,  
        'play.title':music.filename,  
        'play.singer':music.singer,  
        'play.coverImgUrl':music.coverImgUrl,  
        'play.currentTime':'00:00',  
        'play.duration':'00:00',  
        'play.percent':0  
    })  
},
```

播放按钮、暂停按钮、下一曲按钮对应的事件触发

```
// 播放按钮  
  
pPlay: function() {
```

```

        this.audioCtx.play()
        this.setData({
            state: 'running'
        })
    },
    // 暂停按钮
    pPause: function() {
        this.audioCtx.pause()
        this.setData({
            state: 'paused'
        })
    },
    // 下一曲按钮
    pNext: function() {
        var index = this.data.playIndex >= this.data.musicList.length - 1 ? 0 : this.data.playIndex + 1
        this.setMusic(index)
        if (this.data.state === 'running') {
            this.pPlay()
        }
    },

```

(8)播放列表



<!-- 播放列表 -->

```
<view wx:if="{{click==true}}" class="pList">
```

```

    <view class="li_title"><text>播放列表</text></view>

    <image class="close" bindtap="close" src="/pages/images/
index/close.png"></image>
    <view wx:if="{{musicList==''}}" class="no" >

        <text>暂无歌曲</text>

    </view>
    <scroll-view wx:else scroll-y="{{false}}" class="clear"
>
        <view class="pli" wx:for="{{musicList}}" wx:key="id"
bindtap="change" data-index="{{index}}">
            <view class="li_song">
                <text>{{index+1}} {{item.filename}}</text>
                <text wx:if="{{index==playIndex}}" class="playing
">正在播放...</text>

            </view>
        </view>
    </scroll-view>
</view>

```

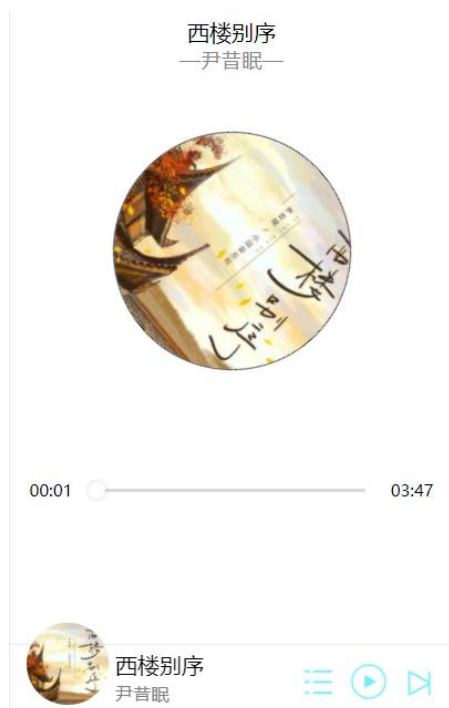
index.js

```

data: {click:false,exit:false}
pList(){
    this.setData({
        click:true
    })
},
exit(){
    this.setData({exit:true})
    console.log(this.data.exit)
},
close(){
    this.setData({click:false,exit:false})
},

```

(9)专辑播放页面



<!-- 专辑显示 -->

```
<view wx:if="{{exit==true}}" bindtap="close" class="content-play">
```

<!-- 显示音乐信息 -->

```
<view class="content-play-info">
  <text>{{play.title}}</text>
  <view>—{{play.singer}}—</view>
</view>
```

<!-- 显示专辑封面 -->

```
<view class="content-play-cover">
  <image src="{{play.coverImgUrl}}" style="animation-play-state:{{state}}" />
</view>
```

<!-- 显示播放进度和时间 -->

```
<view class="content-play-progress">
  <text>{{play.currentTime}}</text>
  <view>
    <slider bindchange="sliderChange" activeColor="#d33a31"
block-size="12" backgroundColor="#dadada" value="{{play.percent}}" />
```

```
</view>  
<text>{{play.duration}}</text>  
</view>  
</view>
```

四、总结

内容只是搭建了一个简易的框架，当然有一些功能我们做成组件来进行复用，同时可以将我们的数据做成一个共享数据池，再者我们调用了较多 api 请求，过多的回调请求会引起“回调地狱”，我们可以对我们的数据请求做一个 Promise 处理。