



**QG工作室项目报告**

学 院 信息工程学院

专 业 电子信息类

班 级 20级 8班

组 别 网络组

姓 名 邓崇昌

学 号 3120002418

2021 年 6月 5日

广东工业大学计算机学院制

**目录**

**云音乐项目**

**1.** **项目简介 3**

**1.1项目背景 3**

**1.2项目基本内容 3**

**2. 设计思路 3**

**2.1项目页面结构 4**

**2.2项目功能结构 5**

**3.** **详细功能设计 6**

**3.1登录注册+用户信息展示和编辑（使用session验证身份） 6**

**3.2轮播图的形式推荐专辑、歌曲 8**

**3.3歌单推荐展示 9**

**3.4按类型将歌曲分类，根据分类筛选歌曲 9**

**3.5模糊搜索歌手、歌曲 10**

**3.6播放音乐、展示正在播放的列表 10**

**3.7插入、删除、清空播放列表 10**

**3.8不使用网页自带的播放器，而是自己制作一个音乐播放器 10**

**3.9制作播放历史 10**

**3.10制作我的收藏歌曲 10**

**3.11制作我的收藏歌单 10**

**3.12上传音乐 10**

**3.13制作歌单 10**

**3.14自定义歌曲分类 10**

**3.15将歌曲添加至播放列表 10**

**3.16收藏播放列表全部歌曲 10**

**3.17在播放列表中或收藏歌单中删除歌曲 10**

**3.18进入歌单使其热度加一，播放歌单使其播放量加一 10**

**4.** **程序测试 11**

**5. 项目亮点 12**

**5.1数据结构 13**

**5.2数据处理 14**

**6. 心得体会 14**

**6.1自我感受 14**

**6.2不足之处 15**QG云音乐项目

**1.项目简介**

**1.1 项目背景**

在当今这个信息网络时代下，生活节奏不断加快，人们的生活压力和工作压力也在不断增加。在这样的背景下，听音乐依靠其在放松心情、减缓压力方面极佳的效果，很快就成功融入进了大多数人的生活方式。据统计，在线听音乐已经成为了绝大部分网络用户上网休闲娱乐的众多选择之一，同现在火热的各大短视频平台及微博一样，受到了众多用户的青睐。由于开发类似的音乐网站不仅可以增加浏览网站的人数，还能带来巨大的经济收益，这一商业趋势使得越来越多的音乐平台兴起，众多音乐软件在互联网上展开了激烈的竞争。

但我认为目前的流行音乐平台中的部分功能模块不一定适用于所有用户，恰好自己现在正在完成工作室的考核，借此机会，个人同时站在用户和开发者的角度，通过参考市面上几大主流音乐平台并结合自己对音乐平台的理解与看法，搭建出了一个能满足用户基本需求并且加上了自己创新元素的音乐网站——QG云音乐项目。

**1.2项目基本内容**

**1.2.1 开发技术**

本项目使用到的开发技术有html、css、js、Node、ejs、express、expression-session、mysql

**1.2.2 开发工具**

本项目使用到的开发工具有VSCode、MySql、HeidiSQL、Navicat

**1.2.3开发目的**

致力于通过此项目加深及巩固先前学过的前端知识，提升自己对相应知识的实际应用能力，学习新的前后端相关知识、清楚前后端是通过何种方式建立联系去实现一个完整功能的，了解整个项目所需经过的流程以及步骤。借此机会参考市面上的主流音乐平台并结合自身想法，努力打造出一个除满足考核要求外，更加贴近用户，增加用户体验且能够满足更多用户实际需求的音乐网站。

**1.2.4基本实现功能**

本网站可提供用户登录注册、个人信息管理、歌曲分类、歌单分类、在线播放音乐、在线搜索歌曲、收藏音乐、收藏歌单、管理播放历史、上传音乐、制作歌单、自定义分类等一系列功能。

**2.设计思路**

**2.1项目页面结构**

本项目将所有页面分为了两部分，一部分为不需要登录即可访问到的主要页面，另一部分是只有用户成功登录后才能访问到的个人中心页面。这样分页面的目的就是让一些要有用户权限才能执行的功能与操作受到一定的保护，增强网站的可靠性与安全性。

**2.1.1 主要页面（非登录访问页面）**

主要页面包括了“首页”、“注册页面”、“登录页面”、“歌曲类别详情页面”、“歌单详情页面”、“全部歌曲类别页面”以及“搜索歌曲详情页面”。这些页面的共性就是与用户信息的相关性与紧密性不强，无需进行登录就可访问以及执行其中的部分功能。

**2.1.2 个人中心（登录访问页面）**

个人中心页面包括了“个人信息页面”、“收藏歌曲页面”、“收藏歌单页面”、“播放历史页面”、“上传音乐页面”、“制作歌单页面”以及“自定义歌曲分类页面”。这些页面的共性是与 “主要页面”恰好相反，这些页面往往与用户的关联更加紧密，有的页面存储的是用户的信息，有的页面展现的是用户最近的操作，有的页面是为了让用户更好的与平台建立联系而设计的……

总而言之，页面及其功能大体的设计思路是从用户的角度出发，一步步围绕着用户、一点点搭建出页面、一步步实现对应功能，最终形成了现在的项目。

下面是项目页面结构流程图：

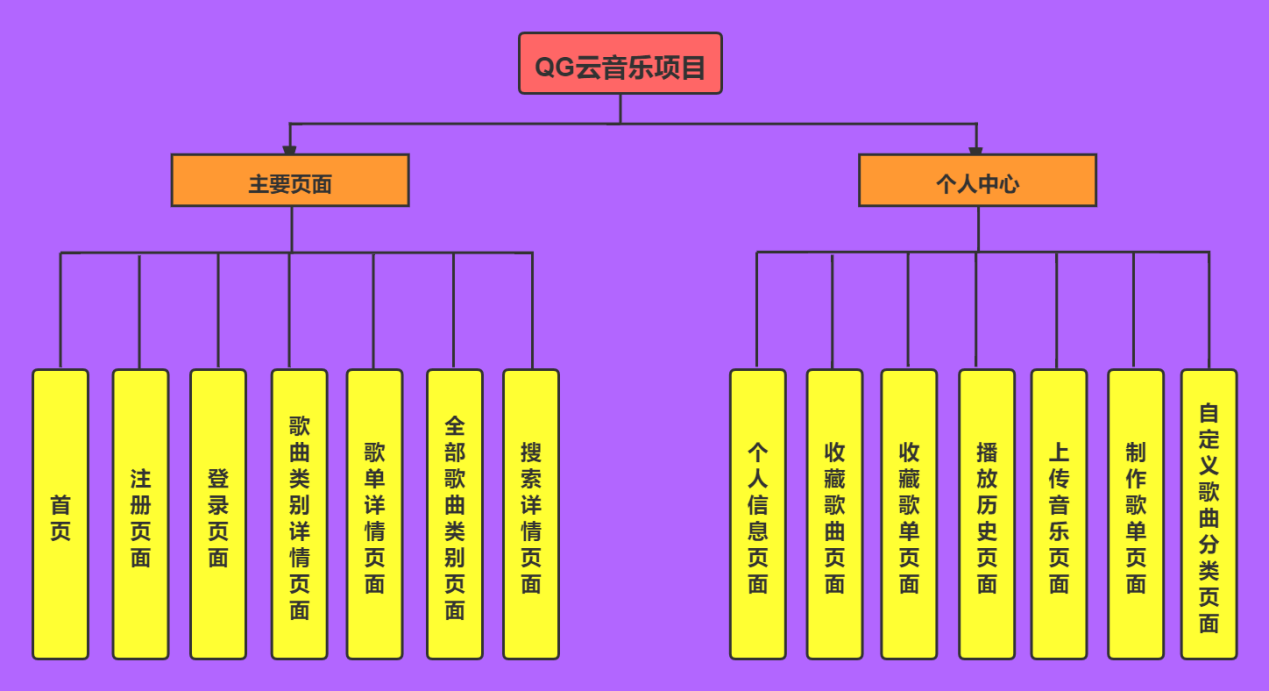
****

图 | 云音乐项目页面结构

**2.2项目功能结构**

随着现在互联网上越来越多的网站被搭建，用户很容易会出现审美疲劳，这也就导致了用户其实并不会在每一个网站都去花费时间注册账号，很多时候用户进入一个网站只是大致的浏览一下。所以，为了使得在该平台注册账号的用户越来越多，就要在用户作为游客浏览网站的时候给予用户良好的体验，才能得到用户不错的反馈，这也是本项目以用户是否登录为分界点将功能分为两部分的原因。

**2.2.1 游客功能**

在用户没有登录的情况下，我们把他们成为游客，他们可以正常的实现大部分音乐平台的功能，例如“对所有歌曲进行操作”、“对轮播图进行操作”、“对播放列表进行操作”、“利用搜索框实现模糊搜索”、“对歌单进行操作”、“对播放器进行操作”、“访问大部分的主要页面”等一系列的功能。

**2.2.2 用户(有账号)功能**

在用户有账号并且成功登录的情况下，他们可以正常的实现音乐平台上所有的功能，例如“管理自己的个人信息”、“管理收藏的歌曲”、“管理收藏的歌单”、“管理播放历史”、“自由上传音乐”、“自由制作歌单”、“自由定义分类”等一系列的功能。

通过这些功能，普通用户不在只能被音乐平台所影响，他们也可以作为创作者去影响音乐平台，将自己的想法结合歌曲和歌单发表至平台让更多的人看到。

下面是项目功能结构流程图：

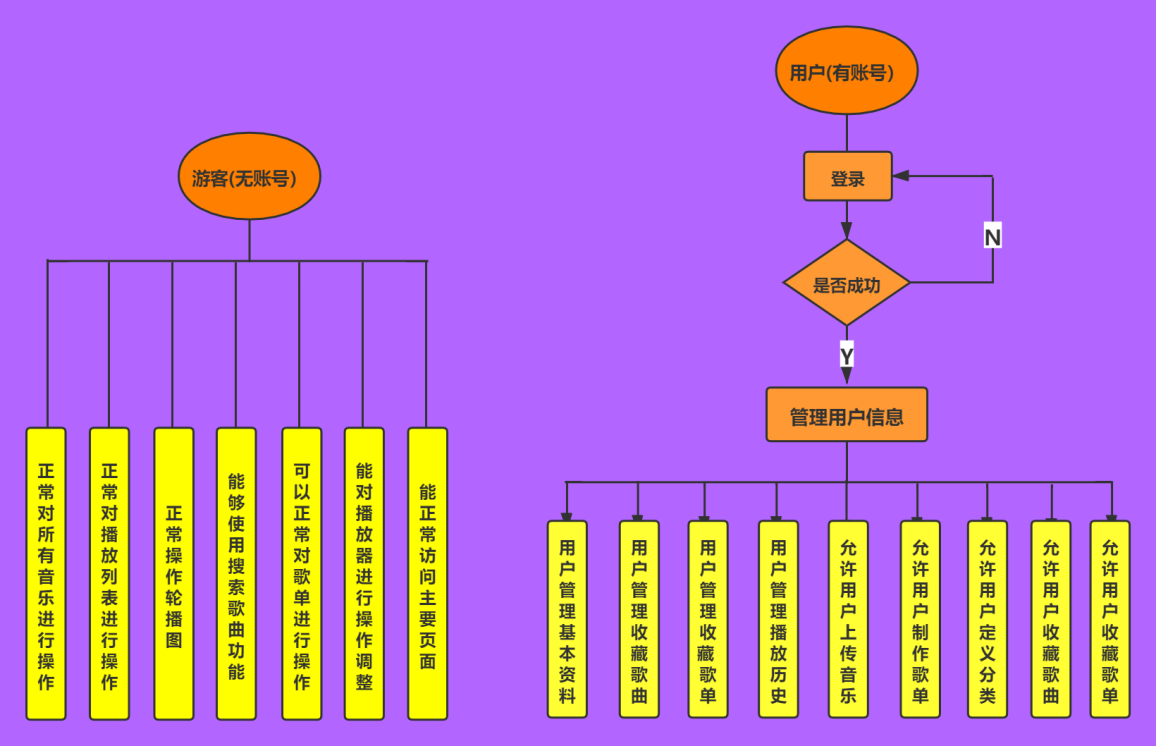
****

图 | 项目功能结构流程图

**3.详细功能设计**

**3.1登录注册+用户信息展示和编辑（使用session验证身份）**

作为面向用户的一个网站，这一类功能是必不可少的，是整个项目的核心入口。这类功能设计的目的一方面是为了让用户更好与音乐平台进行交互，另一方面也是为了方便用户在该平台有自己的权限去管理个人信息和执行发表音乐作品等操作。

**3.1.1 注册功能**

用户在注册账号文本框输入账号后，光标离开该文本框浏览器会利用正则表达式判断账号格式是否正确，若不正确则会给出提示信息，同时浏览器会通过ajax利用get请求到服务端，从数据库中的用户表（存储了注册用户的账号密码）中进行查询，如果出现用户注册账号和用户表里账号一致的情况，服务端会响应提示信息到客户端，用户就可以在注册页面上看到提示信息。

用户在在密码框中输入注册账号的密码后，光标离开该密码框，浏览器会利用正则表达式判断用户密码框中密码的格式是否正确。再用户输入二次密码后，光标离开密码框会将用户输入的确认密码和第一次输入的密码进行比较，若两者不一样会显示提示信息。

账号、密码格式不正确以及账号被注册过都会禁用注册按钮，防止用户提交表单。用户输入信息格式都正确无误后才可以通过注册按钮注册账号，随即浏览器会通过form表单利用post请求方式将账号和密码信息发送至服务端，服务端接收到后会将此消息存储到用户表中，下次用户就可以利用此账号和密码进行登录。注册成功后会跳转至登录页面！

下面是注册功能效果图：



图 | 注册功能效果图（截取自注册页面）

下面是注册账号功能流程图：

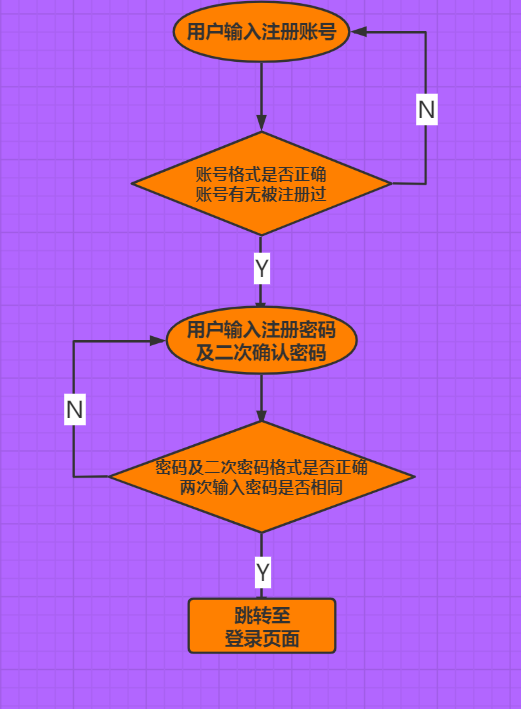
****

图 | 注册账号功能流程图

**3.1.2 登录功能**

用户在注册账号文本框输入账号后，光标离开该文本框浏览器会利用正则表达式判断账号格式是否正确，若不正确浏览器会给出提示信息。用户输入密码后浏览器也是执行类似操作。账号或密码若出现格式不正确的情况，浏览器会自动禁用登录按钮，阻止用户提交表单。

在用户输入的账号密码格式无误并点击登录按钮的情况下，浏览器会通过form表单利用post请求方式将用户的信息发送至服务端，服务端利用在请求体中接受到的信息在用户表中进行查询。若查询不到用户输入的账号密码，则会重定向登录页面并将错误信息返回至登录页面上；若用户表中查询到结果，则服务端会将查询结果也就是用户的账号密码存储到req.session.user中，随即页面会跳转至首页，并会在页面右上角会展示昵称（如果用户还没修改过个人信息则会展示登录账号）。

下面是登录功能效果图：



图 | 登录功能效果图（截取自登录页面）

下面是登录功能流程图：

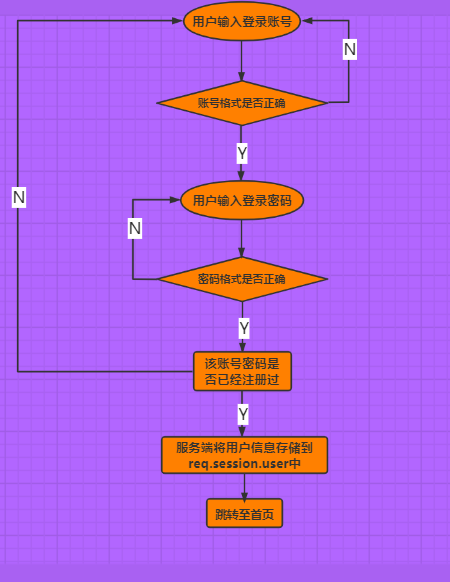
****

图 | 登录功能流程图

**3.1.3 用户信息展示和编辑功能（使用session验证身份）**

根据req.session.user是否为null，可以判断用户有无成功登录，若登录成功，则可以在首页中点击的“我的音乐”或“创作者中心”链接进入个人中心，还可以点击首页右上方的账号昵称，通过下拉菜单中的“个人中心”选项进入个人中心板块。同理，项目中个人中心模块的所有页面都是要req.session.user不为null的情况下才能够打开，收藏歌曲和收藏歌单功能也才能正常使用，若用户在没有登录的情况下点击对应按钮，则会跳转至登录页面提醒用户执行登录操作。

每次加载个人中心模块，服务器都会用ejs模板引擎将一些存储在mysql中的数据渲染至页面中，比如“上次登录时间”、“收藏歌曲数目”、“收藏歌单数目”、“上传音乐数目”、“播放历史数目”以及“上次个人信息修改时间”等等一系列的数据。

页面上有设置“个人信息”模块，用户可在想要编辑对象所对应的文本框中直接进行数据的修改（生日所对应的文本框采取了正则表达式验证格式），修改完成后点击保存按钮后，浏览器会通过ajax将个人信息发送至服务端，服务端通过账号查找到数据库对应的信息，利用sql语句中的update对其进行更新，等到服务端的操作完成后并发送响应后，浏览器会弹出提示信息。如果用户修改了昵称，则右上方的昵称立刻会改变为用户的修改值。由于编辑功能无需跳转页面，所以页面上的信息仍是修改过的信息，下次再次刷新页面时，ejs模板引擎又会根据数据库中的值来渲染页面，呈现出的自然是修改的数据。

下面是个人信息展示与编辑效果图：

****

图 | 个人信息的展示与编辑功能效果图（截取自个人中心页面）

下面是个人信息展示与编辑功能流程图：

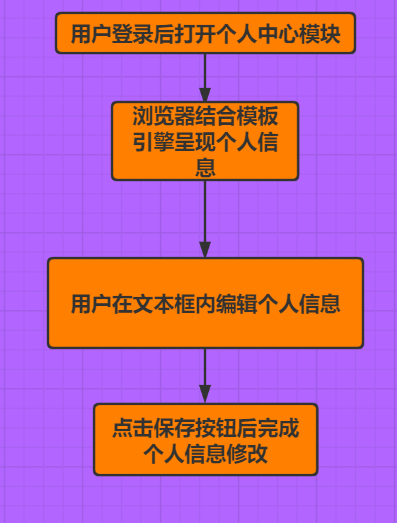


图 | 个人信息的展示与编辑功能流程图

**3.1.4 退出账号功能**

用户可以通过点击昵称引出下拉菜单，在下拉菜单中点击退出后，浏览器通过ajax发送请求至服务端，服务端接受到退出账号的请求后，先将req.session.user的值设为null，这样就恢复至了登录前的状态，随即将退出信息响应至客户端，并渲染登录页面，这样用户就回到了登录页面并能够看到“退出成功”的提示信息。

**3.2 轮播图的形式推荐专辑、歌曲**

现在几乎打开任何一个网站，首先映入眼帘的都是平台的广告轮播图，由于它所在的位置比较醒目特殊，其在网站建设中的地位也在日益上升。个人站在用户的角度考虑后，决定使本项目在大多数音乐网站的轮播图原有功能基础上再添加“点击轮播图后播放音乐”的功能，节约了点击轮播图后再跳转页面的时间。

**3.2.1 轮播图的基础功能**

1、通过定时器的使用结合自己封装好的动画函数实现了轮播图的图片轮播功能

2、通过给动态创建的li添加自定义属性index索引并结合动画函数实现点击轮播图下方的小圆圈跳转至相应轮播图片的功能

3、利用动画函数结合给左右按钮绑定点击事件实现点击左右按钮切换上一张和下一张图片功能

4、鼠标放到轮播图上利用清除定时器自动停止轮播图进行轮播，方便用户观看轮播图详情内容

5、通过在轮播图图片区域ul的末尾克隆第一张图片，并在轮播图滚动到最后一张图片时立即跳转到第一张图片来实现所有图片的无缝轮播滚动

6、创建一个变量flag，让其在动画函数的回调函数内在true和false之间变换，通过判断该变量的值可以达到“节流阀”的目的，使得用户只有在一张图片滚动完时才可以观看到其他的图片，防止用户因过于频繁点击箭头按钮所错过图片信息

下面为轮播图功能模块图片：

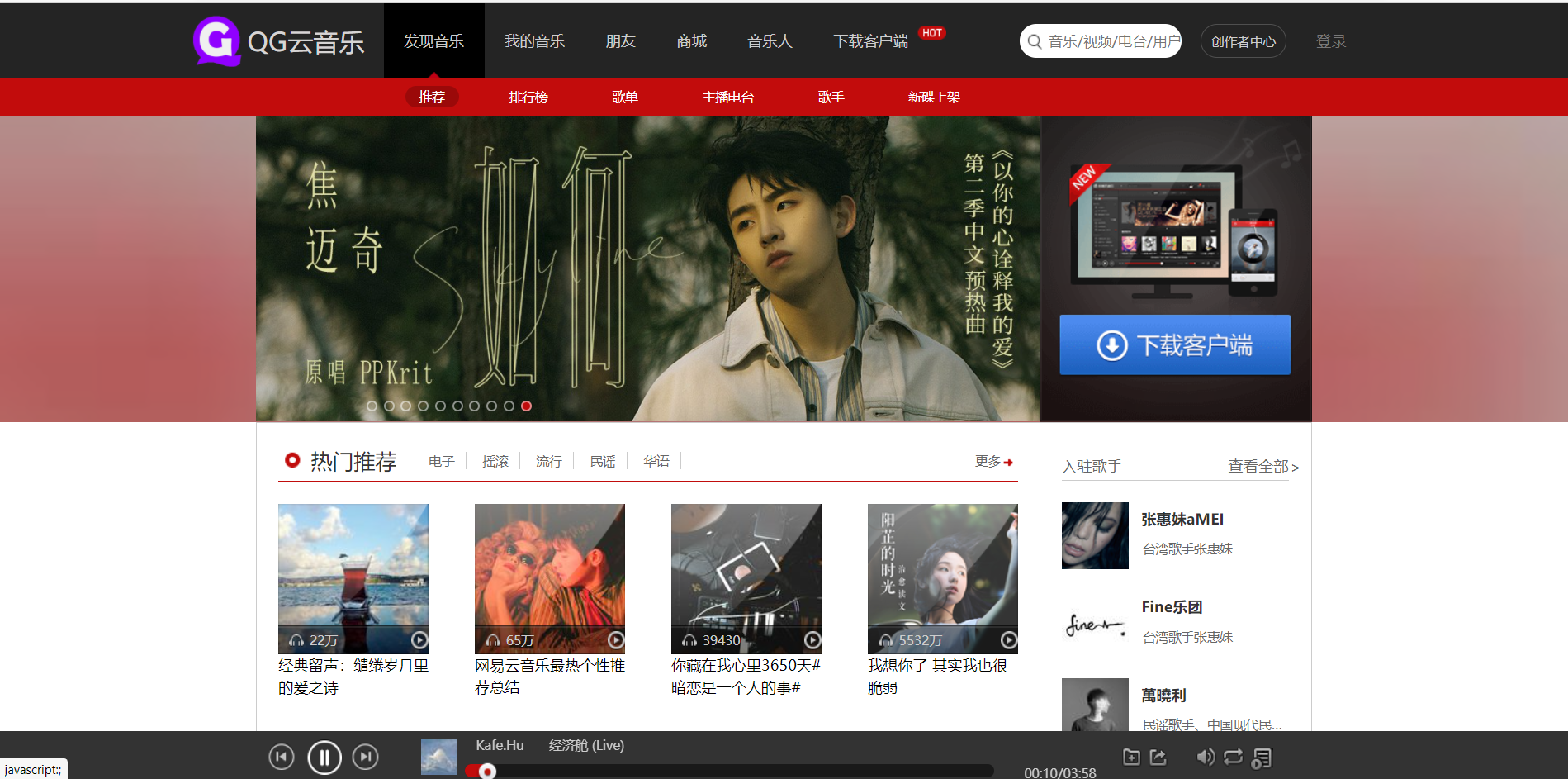


图 | 轮播图功能模块图片（截取自首页）

**3.2.2 轮播图的额外添加功能**

主页面的每个轮播图的图片都装在a链接中，他们是通过ejs模板引擎渲染出来的，在渲染前，服务端会依据轮播图的数量从mysql数据库的所有音乐表中按照热度查询排序靠前的几首歌曲信息，并在渲染时会将歌手名和歌曲名拼接好赋值给a标签的class属性，将他们的音乐编号依次赋值给每个a链接的id属性，如此操作后，每个轮播图对应的a链接中的id值就是其点击后要播放歌曲的编号。

点击轮播图后，浏览器通过ajax将对应图片链接的id值作为数据发送请求到服务端，服务端接受到请求后，根据id值在数据库的全部歌曲表中查询对应的歌曲信息，并将此信息响应至客户端，客户端将服务端响应信息中歌曲和歌手名提取出来，加上前缀后缀拼接成对应音乐的字符串，将此字符串赋值给audio标签的src属性后，通过audio中的各种属性和方法可以对此歌曲进行操作。浏览器接受到响应值后会将歌手、歌曲信息、歌曲总长在播放器中更新，里面的图片也会根据歌曲编号进行相应更换。

点击轮播图后播放的歌曲会自动添加到播放列表和历史记录中，具体操作下文会讲到。有了这个功能后，用户可以很轻松的听到轮播图中展现的歌曲（由于音乐有限，我使用了平台音乐库中热度最高的几首歌曲来代替），增加了用户体验。

下面为点击轮播图播放音乐流程图：

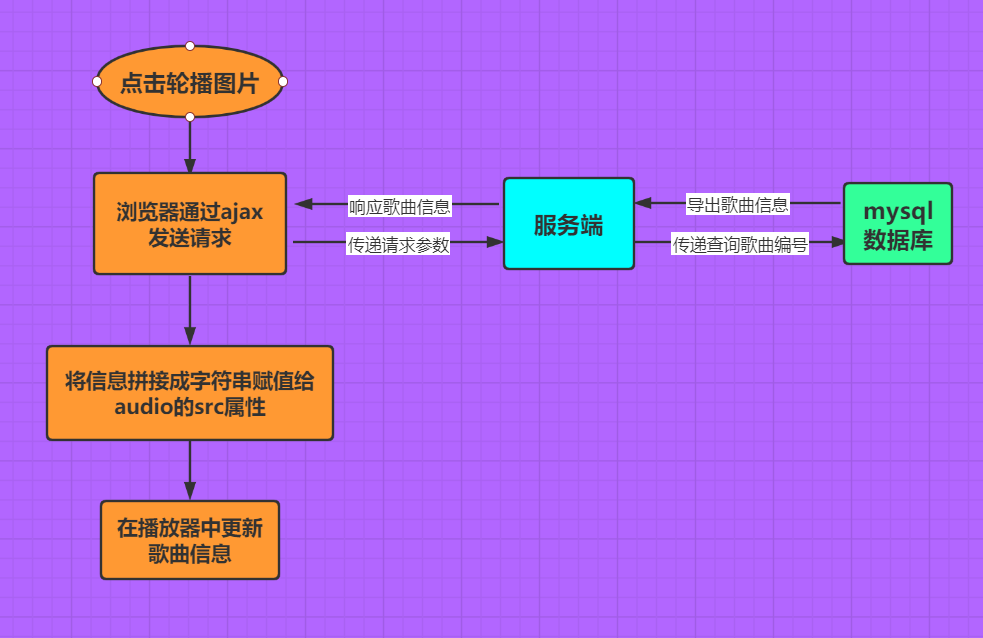


图 | 点击轮播图播放歌曲功能流程图

**3.3 歌单推荐展示**

**3.3.1 热门歌单推荐以及最新歌单推荐**

热门推荐在首页上，每次加载首页时歌单推荐模块都会用到ejs模板引擎进行渲染，渲染前会按照指定的数量从数据库歌单表中依据热度高低排序查询到相信的歌单信息，在渲染的时候就可以使用foreach遍历歌单信息，并给每个要渲染的a链接地址后加上歌单的编号，这样用户就可以通过点击歌单后跳转到此歌单的详情页面了，还要将歌单的信息放置在其对应的具体位置，比如说歌单的封面图需要根据歌单的编号来对应、封面图上的播放量数据是要通过匹配对应歌单信息中的播放量经过处理后得到的、图片下面的歌单标题也是根据在表中查询到的歌单信息中的标题所得到的。

下面为热门歌单推荐功能模块图片：

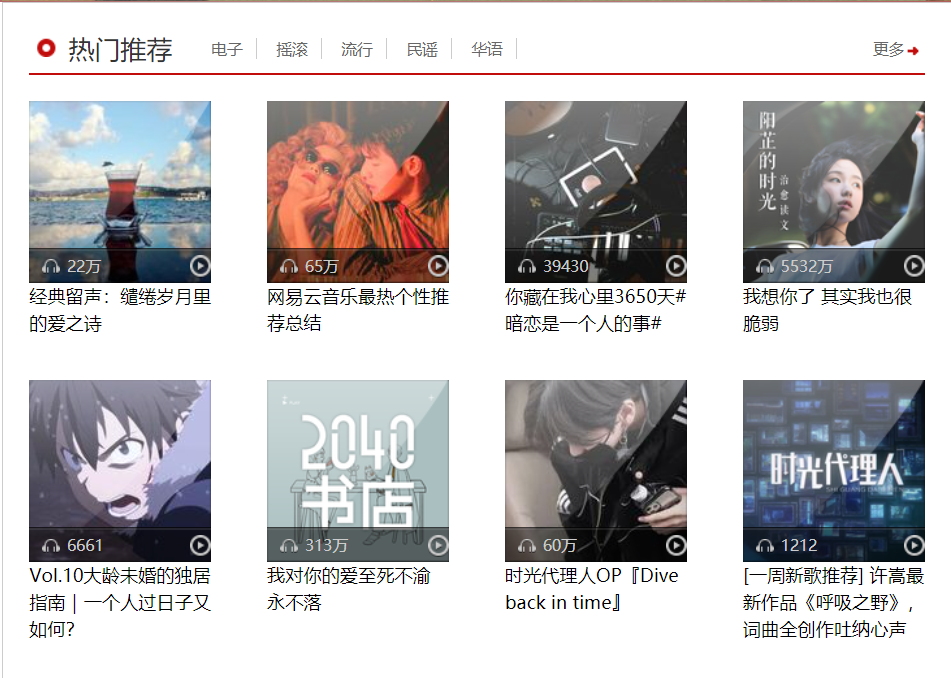
****

图 | 热门歌单推荐功能模块图片（截取自首页）

热门推荐模块下面还有一个最新推荐模块，这一部分推荐的也是歌曲，只不过从数据库中获取歌单信息的条件有所不同，最新推荐是按照指定的数量根据歌单创建的时间从排序后的数据表中查询到的歌单信息，其他方面跟热门推荐类似。

下面是歌单推荐功能流程图：

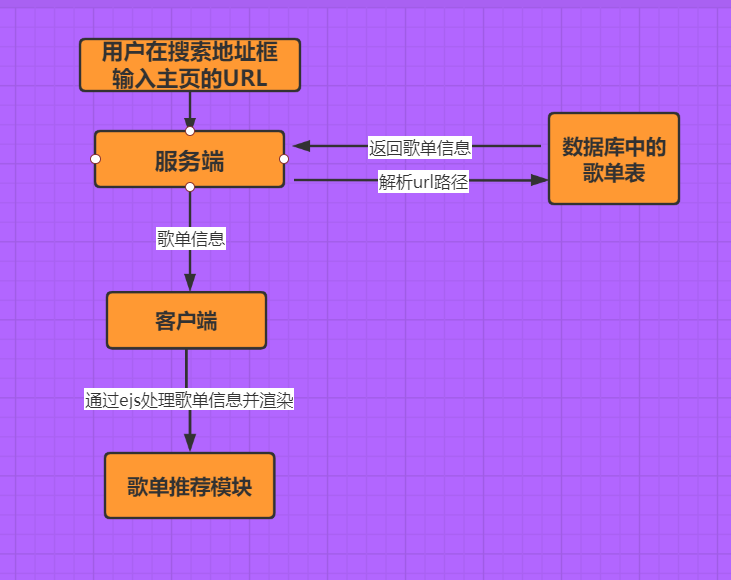


图 |歌单推荐功能流程图

**3.3.2 歌单推荐详情页面**

用户点击相应的链接后，服务端会通过获取url上的信息（歌单的编号），并会在数据库的歌单表中利用编号通过多表查询将总歌曲表与歌单表联系起来，查询到用户点击歌单的信息，包括歌单的标题、播放量、热度、创建时间、歌单里面歌曲的数目等等，并将这些信息渲染到页面上，其次查询到的歌曲也会按照热度排序过后依次渲染到页面上去。

**3.4 按类型将歌曲分类，根据分类筛选歌曲**

设计歌曲分类可以更好的将歌曲划分为几大模块，便于用户更轻松的选择自己喜欢的歌曲，也有利于后台人员区管理和维护。

**3.4.1 热门歌曲分类以及最新歌曲分类**

每次浏览器加载首页时，服务端会按照指定的个数从数据库歌曲分类表中利用sql语句按照歌曲分类的热度或者该分类创建的时间排名筛选出对应的歌曲类别名称，随后在渲染首页的时候通过ejs模板引擎foreach循环按数据的顺序将类别名称渲染在页面上，并且要使得歌曲类别模块模板中的a链接最后的数字对应其歌曲类别详情页面的编号，以至于用户在主页点击这些歌曲分类名称后能够跳转至相应歌曲类别详情页面。

下面为歌曲分类功能模块流程图：

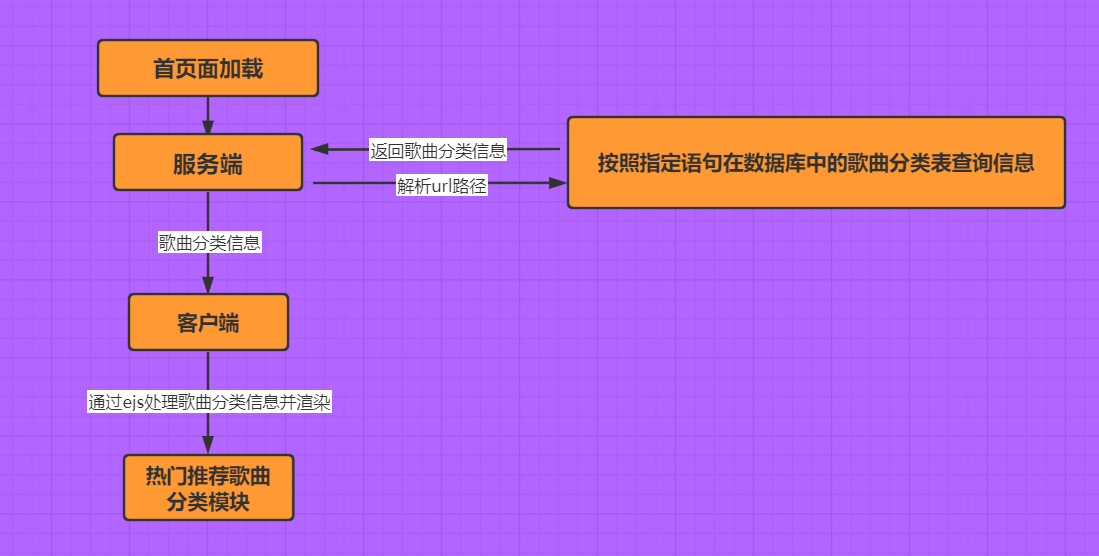


图 | 歌曲分类功能模块流程图

**3.4.2 全部歌曲分类详情页**

跟上述渲染的方法很相近，区别在于服务端通过sql语句查询到了所有的歌曲，然后再通过ejs循环渲染模板的方法使所有歌曲分类呈现在页面上，文字的链接也要对应其歌曲类别详情页面的编号，以便于用户可以通过点击文字链接跳转到对应歌曲类别详情页。

**3.4.3 歌曲分类详情页中的歌曲展示**

在用户点击了歌曲类别链接之后，由于前面模板的设置，url上的末尾会有对应类别的编号，服务端通过sql语句将全部歌曲表与url的编号联系起来，通过查询条件查询出对应的歌曲，按照歌曲的热度将其排序后依次渲染到页面上。

**3.5 模糊搜素歌手、歌曲**

身处在信息化时代，每一个网站上都有着成千上万条信息，这是，模糊搜索这个功能就有着至关重要的作用，用户只需通过在搜索框中输入关键字，就可以快速地查询到其想要的信息，极大程度地增加了用户的体验、提高了用户浏览网站的质量。

**3.5.1 用户在搜索框输入查询信息后下方出现提示信息**

给搜索框绑定了input监听事件,用户在搜索框中输入或删除时都会触发此事件，触发后浏览器会先获取到搜索框中的值，并且设置定时器在指定时间后通过ajax携带着请求参数（用户在搜索框中输入的关键字）发送请求到服务端，设置定时器的目的是为了避免每次搜索框内容变动后浏览器都会发送太多次鸡肋的ajax请求。服务端可以在req上面获取到该关键字、然后在数据库中通过sql语句的“like”在全部音乐表中进行模糊搜索，查询到歌曲信息按照后，服务端会将查询到的信息响应至客户端。

客户端利用响应会的信息依靠响应回来的数组foreach循环拼接成一个字符串，在循环结束后，将拼接好的字符串赋值给搜索框下方的ul.innerhtml，并让ul显示出来，这样用户就可以在不跳转页面的情况下看到该关键字所对应的歌曲信息。

下图为输入搜索框下方提示信息功能图片：

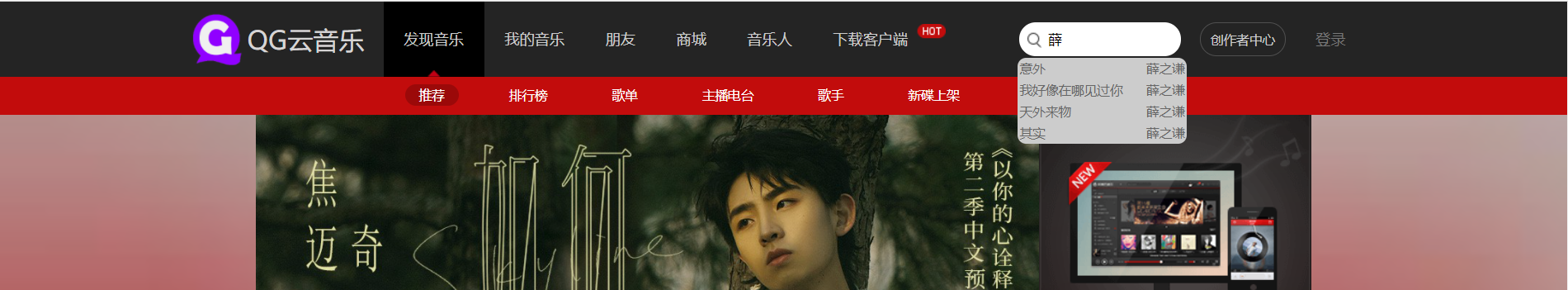
****

图 |输入搜索框下方提示信息功能图片（截取自主页）

**3.5.2 用户点击搜索按钮进入搜索歌曲详情页面**

用户在搜索框输入关键字后，可以通过在键盘上按下回车键（Enter）或者点击页面中的搜索图标，点击后浏览器会通过改变location.href并带上后缀（搜索框中的关键字），服务端在req中可以获得到关键字并在全部歌曲表中使用该关键字进行模糊查询，将查询到的所有歌曲按照歌曲热度排序好后使用ejs模板引擎渲染到搜索详情页面上去，这样用户就能在页面上看到该关键字查找到的所有歌曲信息了。

**3.5.3 用户点击搜索框下方的提示信息进入搜索歌曲详情页面**

每个提示信息的父盒子都是a标签，给这些链接绑定点击事件，每当用户点击了之后，执行的操作相当于是用户在搜索框中输入了搜索信息中的歌名，随即点击了搜索按钮，这样用户在点击下方的提示信息之后，就会跳转到搜索歌曲详情页面，而这个页面里面就只有用户点击的提示信息所对应的那一首歌曲。

下面是搜索功能模块流程图：

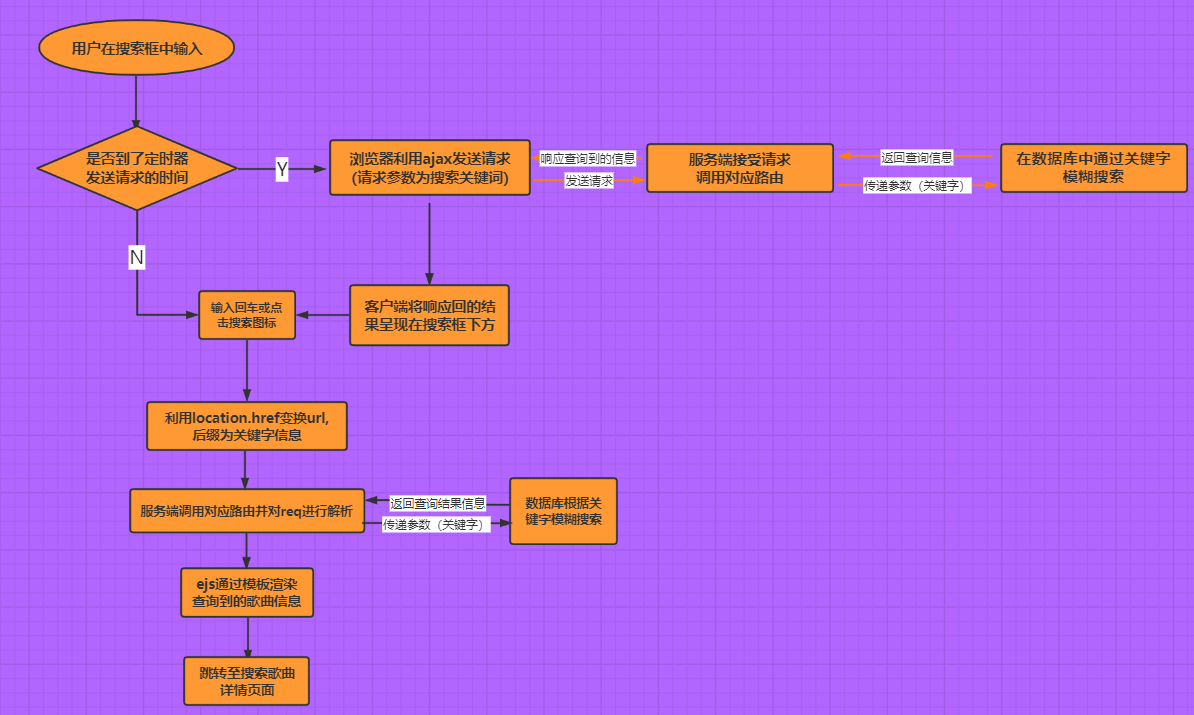
****

图 | 搜索功能模块流程图

**3.6 播放音乐，展示正在播放的列表**

这一功能是作为一个音乐网站最重要，也是最核心的功能。

**3.6.1 点击轮播图播放音乐**

这一功能在前面讲述轮播图功能的时候有提及过，由于每个装有轮播图图片的a标签的class属性值以被渲染为歌手名称加歌曲名称的形式，所以在用户点击轮播图片之后，浏览器只需将准备好的字符串模板和class属性值拼接起来再赋值给audio标签的src属性，随即通过使用audio的属性和方法，比如说audio.play，audio.paused等等就可以对正在播放的音乐进行相应的操作了。

**3.6.2 点击每个音乐所对应的播放按钮来播放音乐**

每一个播放按钮都是通过ejs渲染到页面中的，渲染时用的模板里面就有让播放按钮的id值等于其对应歌曲的编号，每次点击播放按钮之后，浏览器会先获取到点击按钮的id值，随即通过ajax向对应地址发送请求，请求参数就是点击按钮的id值，服务端接受到请求后会从req获取到该请求参数，将请求参数作为查询函数的参数在数据库的全部歌曲表中查询到对应的歌曲信息，然后将此信息响应至客户端，客户端接受到之后将信息中的歌曲名称和歌手名称剖析出来，跟指定的模板组合成一个符合audio.src格式的字符串，之后将其赋值给audio的src属性，随即通过使用audio的属性和方法，比如说audio.play，audio.paused等等就可以对正在播放的音乐进行相应的操作了。

**3.6.3 播放音乐后将播放歌曲添加至播放列表和历史记录中**

在点击播放按钮后，客户端可以接受到响应回来的对应歌曲编号下的歌曲信息，然后剖析出歌曲信息中的歌手和歌曲名，让其和播放列表中li里的模板字符串拼接起来，并且让每个最新添加的歌曲信息相对应的字符串部分放置在总字符串的最前面，随后再将拼接好的总字符串赋值给播放列表的ul，这样在打开播放列表模块后，就可以看到刚刚播放歌曲的信息已成功的添加至了播放列表。

不过如果仅仅是这样操作，确实可以在不刷新页面的情况下持续添加歌曲到页面中的播放列表中，一旦刷新页面后，之前添加在播放列表的歌曲将全部消失，因为每次刷新页面，模板引擎渲染播放列表模块的时候都需要循环遍历数据库中播放列表这个表中的值，之前的操作并没有将歌曲信息添加到该表中，所以自然什么也渲染不出来，用户刷新后看到的是空空如也的播放列表。

所以服务端根据请求参数值（歌曲编号）查询到对应的歌曲信息后，还需要执行其他的操作，首先，在服务端需要调用一个已经写好的时间函数，用于获取现在的时间，方便以后查询数据库中播放列表这个表时可以按照时间将所以的歌曲按照一定的顺序排列好，除了获取时间的函数，还需要一个能够将数据存储（歌曲编号、歌曲名称、歌手名称、播放时间）到播放列表中的函数，下次需要渲染页面的时候在模板中依次添加的是按照播放时间从数据库播放列表中查询到的歌曲，这样就保证了刷新页面后用户依然可以看到之前播放的歌曲，而且由于播放列表中的歌曲是按照播放时间排序渲染出来的，最近一次播放的歌曲会在最上面，符合绝大多数用户的使用准则。

为了方便后续实现管理播放历史的功能，所以在服务端将歌曲信息添加到数据库播放列表后，还需要一个能够将歌曲信息添加到历史记录表中的函数（参数依然为歌曲编号、歌曲名称、歌手名称、播放时间），记录下时间的目的是为了用户查看历史记录的时候也可以看到自己是什么时候听的歌曲，其余大致操作和将歌曲信息添加至播放列表几乎相同。

下面是点击轮播图播放音乐以及将音乐信息存储到播放列表和历史记录的流程图：

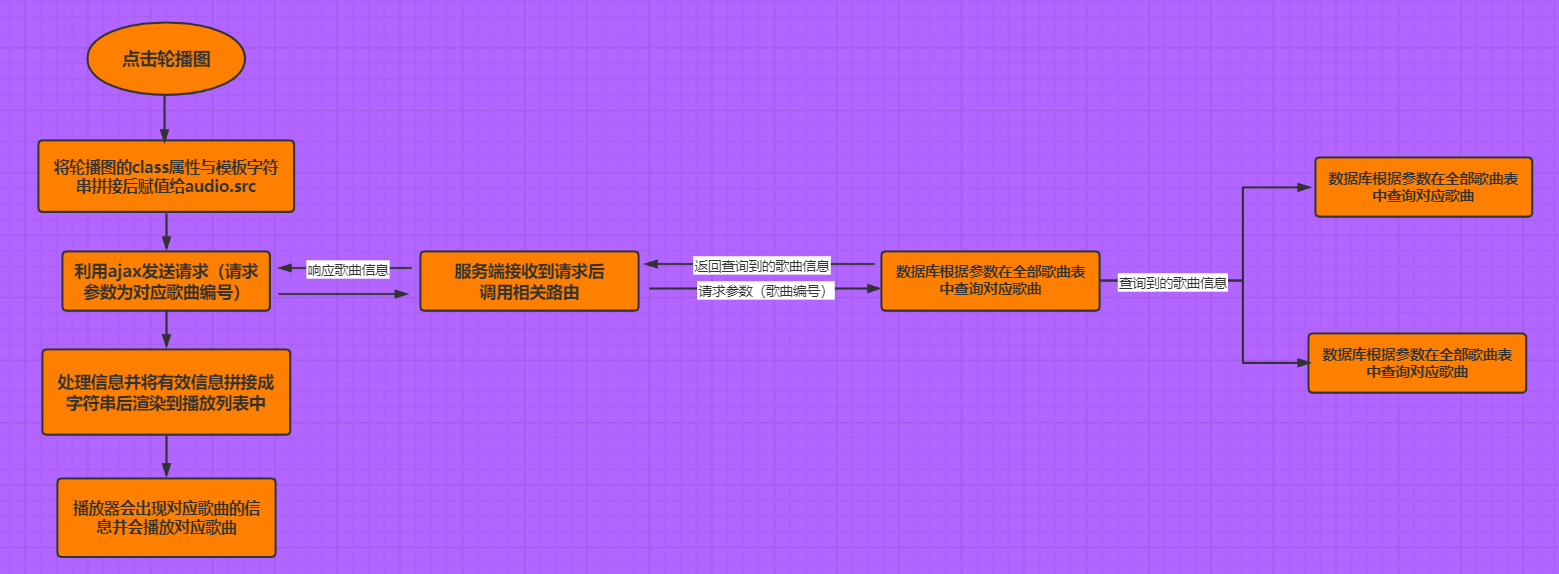
****

图 | 点击轮播图播放音乐以及将音乐信息存储到播放列表和历史记录的流程图

下面是点击播放按钮播放音乐以及将音乐信息存储到播放列表和历史记录的流程图：

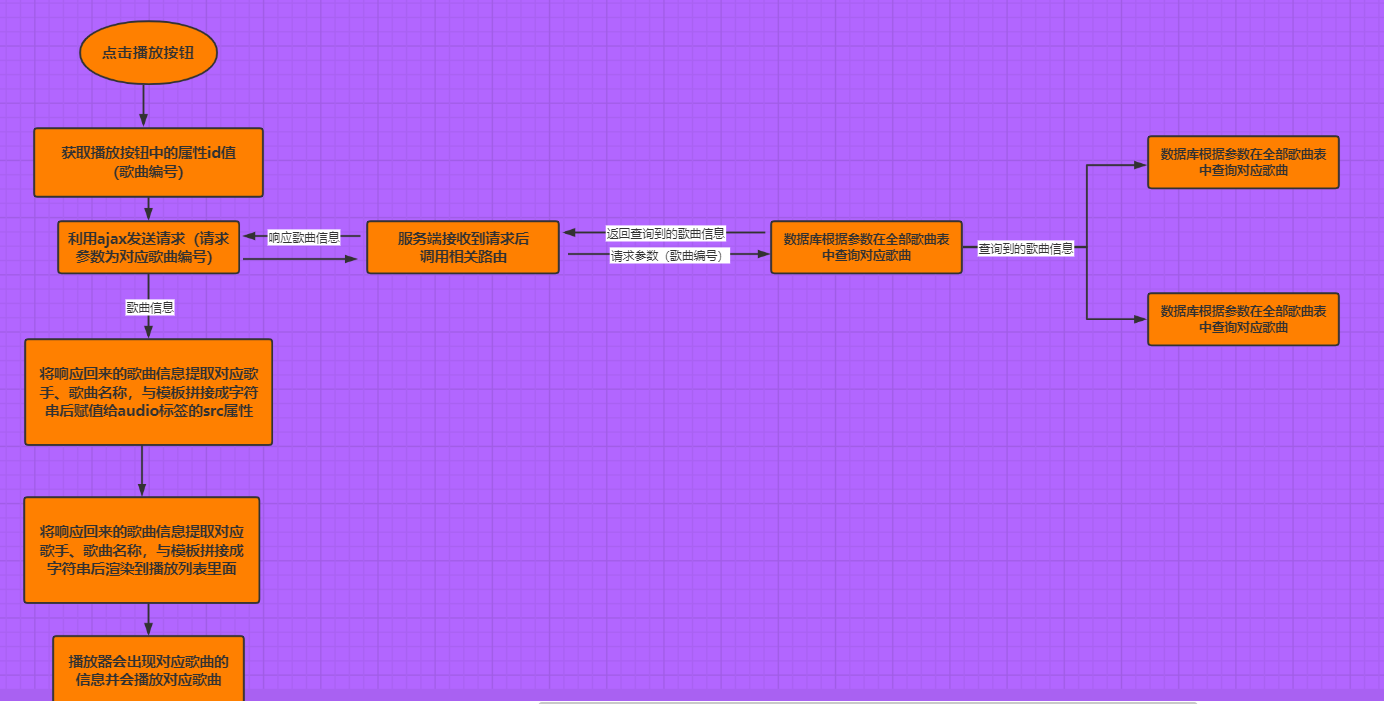
****

图 | 点击播放按钮播放对应音乐以及将音乐信息存储到播放列表和历史记录的流程图

**3.7 插入、清空播放列表**

基本所有的音乐网站都配备了这一项功能，因为其在极大程度上能够帮助用户管理播放列表，用户也得以在短时间内高效地管理音乐。

**3.7.1 清空播放列表**

在播放列表的右上方有一个“清除”链接，用户点击了此链接后，浏览器会使播放列表所对应的ul.innerhtml为空，从而实现页面上播放列表为空的效果，但刷新之后播放列表又会恢复到原来的样子，所以要想在刷新后播放列表中没有歌曲，就必须要清除数据库的播放列表。点击链接后浏览器会对响应地址发送ajax请求，服务端接收到之后会调用一个能够清空数据库播放列表的函数，此函数利用的是sql语句中的truncate去清空对应数据表（播放列表），这样才算真正的清空了播放列表。

下面是清空播放列表流程图：

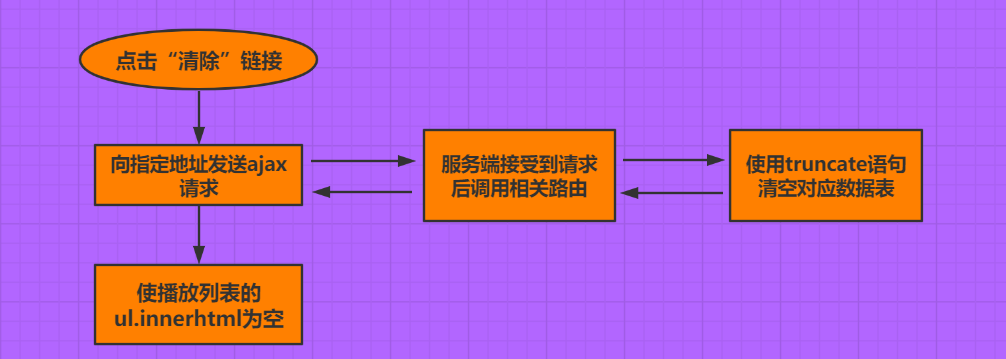
****

图 | 清空播放列表功能流程图

**3.7.2 插入播放列表**

当用户进入到歌单详情页面时，在歌单所有音乐的上方，有一个播放全部按钮，用户点击此按钮后，浏览器会对当前的网址url进行解析从而获取到歌单的编号，之后通过ajax发送请求，请求参数就是获取到的歌单编号，服务端接收到请求后，调用相关的路由，执行函数使用数据库通过多表查询的方式依靠歌单编号查找到对应歌单中的所有歌曲，查询信息过后会作为响应值传递到客户端，浏览器接受到响应后，第一步先解析响应信息，获得到歌单第一首歌曲的信息并将其歌手与歌曲名称与指定模板拼接好后赋值给audio的src属性，这样就实现了点击“播放全部按钮”后立即播放其歌单的第一首歌曲功能，这也是现在各种流行平台的呈现效果。第二步先将播放列表清空，随即再以解析响应信息并拼接字符串模板的方式，将查询到的歌单中的歌曲依次渲染到页面上的播放列表中，这样就实现了用户在打开播放列表之后，能够看到了歌单里的歌曲。

若仅仅是做了上述操作，则在用户刷新之后，刚刚加载到播放列表的歌曲又会恢复到之前的样子，因为上述操作只是对页面进行了操作，没有对数据库中相应的表实施清空或者是添加数据，所以要想在刷新页面后仍保持着“播放全部”所呈现的效果，服务端就需要在获取歌单所有歌曲信息后，执行两个函数，一是对数据库播放列表利用sql语句清空里面所有的歌曲内容的函数，二是随后再循环遍历查询到的歌曲信息，将他们依次添加到数据库播放列表中去，这样下次刷新页面时会先从数据库中获取相应数据，播放列表就可以保持着歌单里的歌曲。

下面是插入播放列表功能流程图：

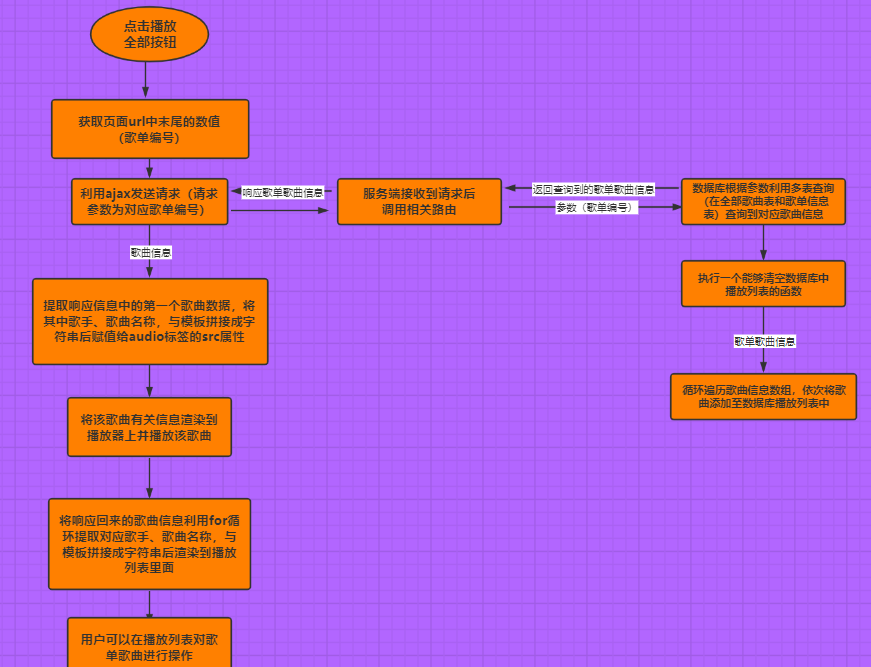
****

图 | 插入播放列表功能流程图

**3.8 不使用网页自带的播放器，而是制作一个自己的音乐播放器**

其实audio标签是可以提供一个播放器用来操作正在播放的歌曲，但是因为其不够美观，功能也受到了一定的限制，为了让用户能够体验到更多的功能，所以最终还是决定自己去做一个音乐播放器来完成对音乐的基本操作。

**3.8.1 显示正在播放歌曲信息**

在用户执行每一次播放歌曲操作的时候，浏览器都会将正在播放所对应歌曲的信息进行解析，将里面的歌手和歌曲名称渲染到播放器对应的位置上去，歌曲的图片是根据正在播放音乐的编号来变换的，编号就是对应图片链接的尾号，但由于没有存储那么多的图片，所以后面采用了为编号取余的方式来加载图片。为了保证刷新页面后播放列表中仍然还有播放信息呈现出来，每一次播放歌曲时还要在服务端多执行一步操作，把正在播放歌曲的信息利用sql语句update更新数据库中上一次播放音乐表，这样每次刷新页面渲染到播放器模块时都会从这张表中先获取所需信息。

**3.8.2 暂停和继续播放功能以及进度条随着播放时间自动变化功能**

当audio.src有值之后，浏览器就可以通过audio的属性和方法来控制当前正在播放的歌曲，播放器右侧立即会通过audio的属性duration获取到歌曲总时长并将其呈现在页面上，将这些属性的变换与播放器中的暂停和播放按钮的点击事件绑定起来，用audio.paused可以监听正在播放音乐的状态（暂停还是正在播放），点击按钮时如果音乐正处于播放状态，那就让图片切换成暂停按钮，并且使用audio.pause()方法暂停音乐播放。同理，如果击按钮时如果音乐正处于暂停状态，那就让图片切换成播放按钮，并且使用audio.play()方法继续播放音乐。这样就实现了控制当前播放音乐的暂停和播放功能。

结合定时器和动画函数，用进度条的总长度和每个音乐的总时长算出每一秒子进度条的位移，再与子进度条的宽度相加，这样就实现了进度条随着音乐进度滚动的效果。并且在动画函数里面，需要获取到当前音乐正在播放的时间，单位是s，需要先转换为几分几秒的形式，然后呈现在进度条的右侧，结合定时器之后，这样音乐每播放一秒，进度条右侧展示的音乐现在播放时间都会发生变化。

播放按钮和进度条需要结合起来，即点击播放按钮时，开启定时器使进度条和正在播放时间变化；点击暂停按钮后，关闭定时器使进度条和定时器停止变化。

下面是暂停和继续播放功能以及进度条随着播放时间自动变化功能流程图：

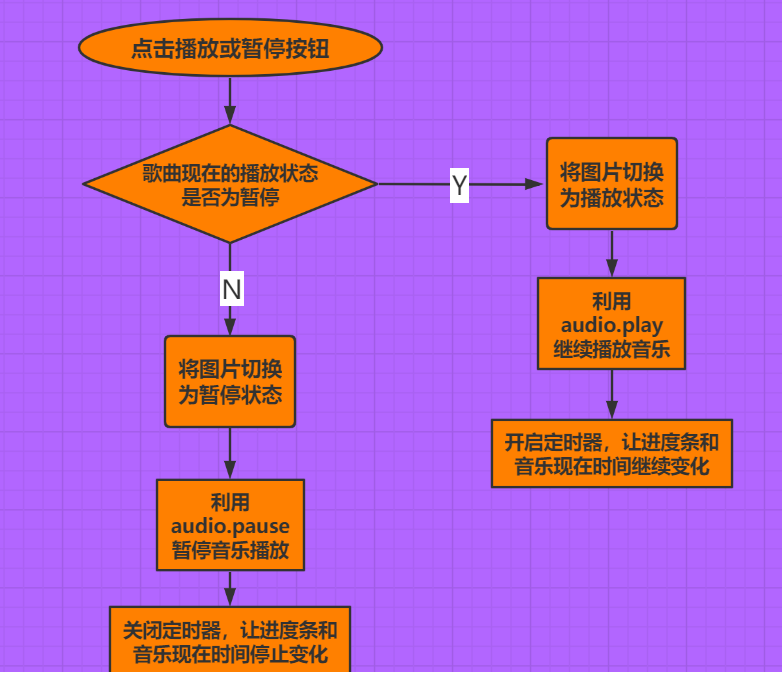


图 |暂停和继续播放功能以及进度条随着播放时间自动变化功能流程图

**3.8.3 点击或拖动进度条实现音乐进度改变**

给进度条绑定了“mousedown”事件，通过获取鼠标点击的位置信息，改变进度条的宽度，让进度条先跳至鼠标点击的位置。给进度条还要绑定“mouseup”事件，当用户鼠标弹起的时候，迅速获取到进度条当前的位置信息，根据和整个进度条以及播放音乐总时长的比例，调整正在播放音乐的音乐进度，并且让进度条右侧的正在播放时间发生与之对应的改变。

在绑定的“mousedown”事件里，还document绑定一个“mousemove”事件，在里面也调用了根据鼠标位置改变进度条的函数，使用户能够通过拖动进度条的方法自由改变进度条的位置以及音乐正在播放的时间。

**3.8.4 切换当前播放歌曲的上一首或者下一首歌曲**

在播放和暂停按钮两边，分布着切换至上一首歌曲和下一首歌曲的按钮。每个页面的播放器模块都是结合“上次播放歌曲表”中的信息渲染出来的，在模板中给每个按钮都添加一个id属性，渲染的时候只需将表中的歌曲编号赋值给id值即可。随后再点击“上一首音乐”按钮之后，浏览器先获取点击按钮的id值（正在播放歌曲的编号），随即通过ajax发送请求到相应的地址，请求参数就是该按钮的id值，服务端调用对应的路由并执行在数据库的播放列表中按照时间顺序查询上一首播放的歌曲，随后将查询结果响应至客户端。客户端通过判断响应结果是否为空，若为空，则弹出警示框表明“您目前播放的音乐已经是最前一首音乐了”；若不为空，则浏览器会根据响应值解析出歌手和歌曲名并与模板字符串拼接后赋值给audio的src属性，顺便会将解析出来的歌手、歌曲名渲染到播放器响应的位置，之后使用其load和oncanplay方法获取该歌曲的总长并渲染到页面上，播放器各按钮的id值和歌曲图片也要根据该歌曲的编号有所变动。切换至下一首歌曲操作与之相似，不做过多赘述。

下面是切换当前播放歌曲的上一首或者下一首歌曲功能流程图：

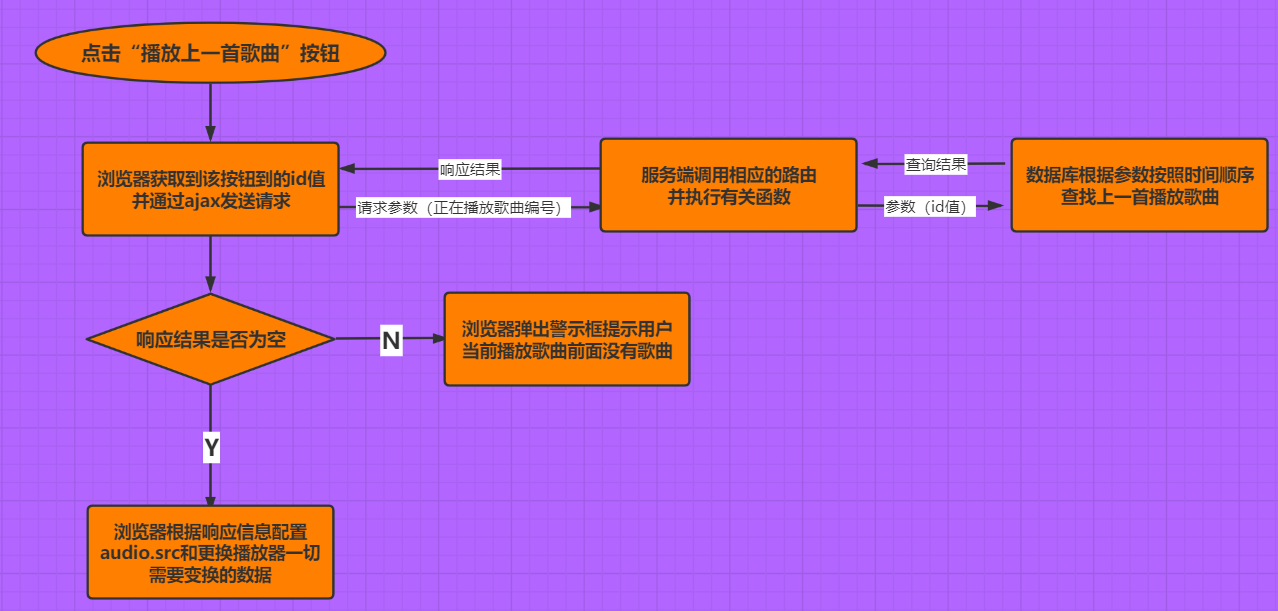
****

图 |切换当前播放歌曲的上一首或者下一首歌曲功能流程图

**3.8.5 点击“收藏”按钮将正在播放的歌曲添加至我的收藏当中**

“收藏按钮”按钮在渲染时也被加上了id值，在点击“收藏”按钮之后，浏览器会立即获取该id值也就是正在播放歌曲的编号，随即利用ajax 发送请求到服务端，请求参数就是歌曲编号，服务端接收到之后调用对应路由，利用中间件middleware的对象封装需要用到的model层的函数，首先获取到传递过来的参数（歌曲编号），在全部歌曲表中先查询到对应的歌曲信息，随后将此歌曲信息的有效信息（歌手名、歌曲名、歌曲编号等）解析出来结合当下的时间time一起作为参数传递给一个能够将数据参数存储在我的收藏表中的函数，执行完毕后，发送响应值到客户端，前端接受到之后以警示框的形式提示用户收藏成功！

这样操作确实可以成功收藏歌曲，但是并不能带给用户最佳体验，如果用户忘记了自己之前收藏过一首歌曲，而他现在又重复收藏了同样的歌曲，用户的收藏列表中就会出现两个收藏时间段不同但其他均相同的歌曲。当我站在用户的角度，我会觉得无论是视觉感受还是使用起来的便捷程度都显得太过臃肿。所以自己在原先的步骤上面添加了一个先在原先我的收藏表中删除该歌曲信息的操作。

也就是在执行往数据库我的收藏表添加歌曲前，需要一个函数用来在我的收藏表中先删除该歌曲，执行完删除操作之后在then回调里面再去调用将歌曲添加至我的收藏表中的函数，这样如果用户在之前有收藏过一首同样的歌曲，用户重复收藏该歌曲后，原先的收藏歌曲会被消除，重复收藏的歌曲会出现在最前整个收藏列表中的最前列，显示的收藏时间也是最新一次收藏的时间。

下面是收藏歌曲功能对应流程图：

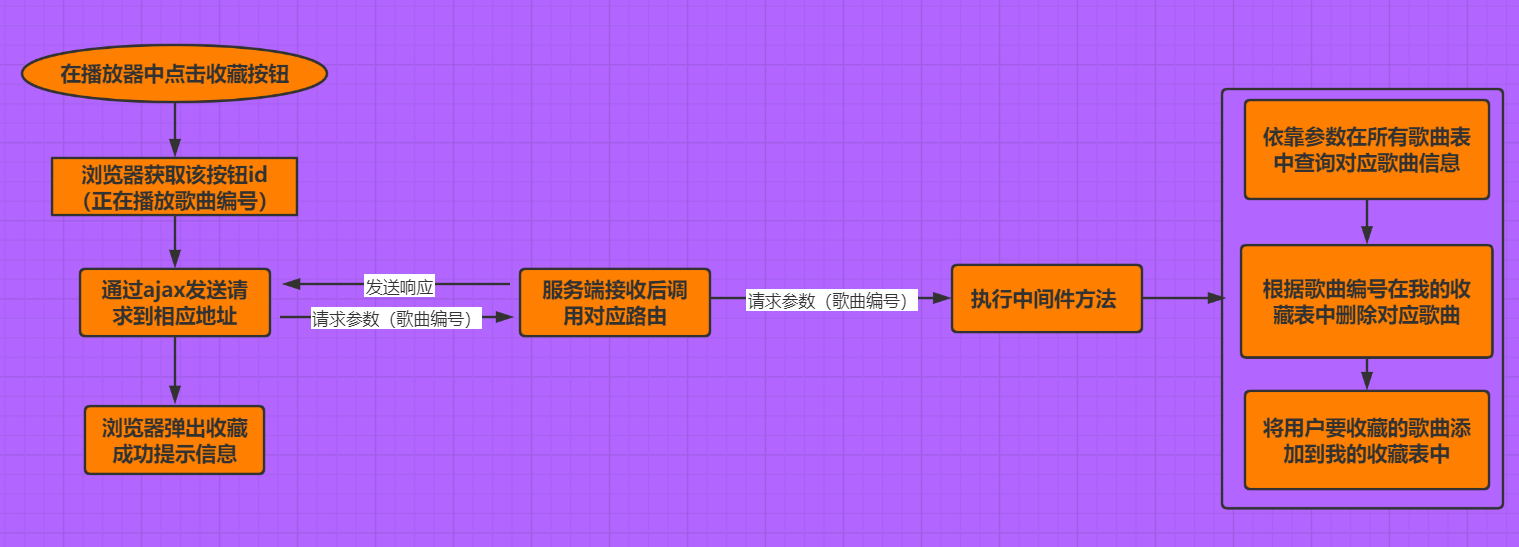


图 | 收藏歌曲功能对应流程图

**3.9 制作播放历史**

由于之前所有能够播放歌曲的操作都会将对应歌曲以及播放时间添加至数据库播放历史记录中去，比如说点击轮播图播放音乐会将轮播图对应的音乐添加至历史记录中，点击播放按钮播放音乐也会使对应音乐添加至历史记录中，还有播放器中的切换上一首或者是下一首歌曲操作，也都会使得对应歌曲添加至历史记录中。所以在渲染我的播放历史页面上，服务端需要将数据库播放历史表中的数据按照时间顺序排序好后，交给ejs模板引擎渲染后即可看到用户的播放历史记录。

下面为制作播放历史功能流程图：

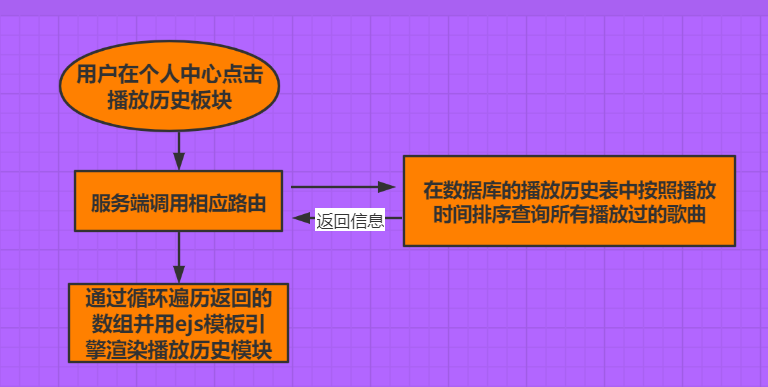
****

图 | 制作播放历史功能流程图

**3.10 制作我的收藏歌曲**

在本项目中，几乎能够播放歌曲的地方都能够收藏歌曲，无论是在播放器中、播放列表中、歌曲分类详情页面中、歌单详情页面中、搜索详情页面中还是播放历史详情页面等等，收藏功能无处不在，但最终的目的都是将对应的歌曲信息以及收藏时间存储到数据库我的收藏表中，所以在渲染我的收藏歌曲页面前，服务端需要先从我的收藏表中按照收藏的时间排序后将数据通过ejs模板引擎渲染到我的收藏模块中去，这样用户就可以在页面上看到我的收藏歌曲了。

下面为制作我的收藏歌曲功能流程图：

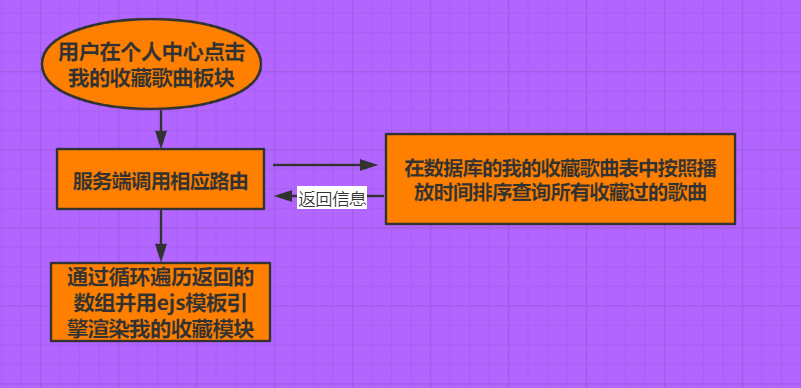


图 | 制作我的收藏歌曲功能流程图

**3.11 制作我的收藏歌单**

在歌单详情页中有个“添加至我的收藏”按钮，点击之后浏览器会先获取到location.pathname的末尾数字（即对应歌单的编号），之后通过ajax发送请求到对应地址，请求参数为刚刚获取到的歌单编号，服务端通过地址去调用不同的路由，使用对应中间件中的方法来执行操作，在中间件的方法中首先要将req中的id值（前端传来的歌单编号）作为参数传递给查询歌单信息的函数，在这个函数的then回调中，需要先根据歌单编号在我的收藏歌单表中删除对应的歌单（此处操作目的与收藏歌曲类似，避免出现用户收藏多个相同的歌单），随后将查询到的歌单信息（歌单编号、歌单标题、播放量、热度等等）以及收藏时间一起作为参数传递给一个能够将信息存储到我的收藏歌单表中的函数，中间件这个方法执行完毕之后，服务端会发送响应至前端，浏览器也会弹出“该歌单已成功添加至您的收藏歌单中”。

下面为制作收藏歌单功能流程图：

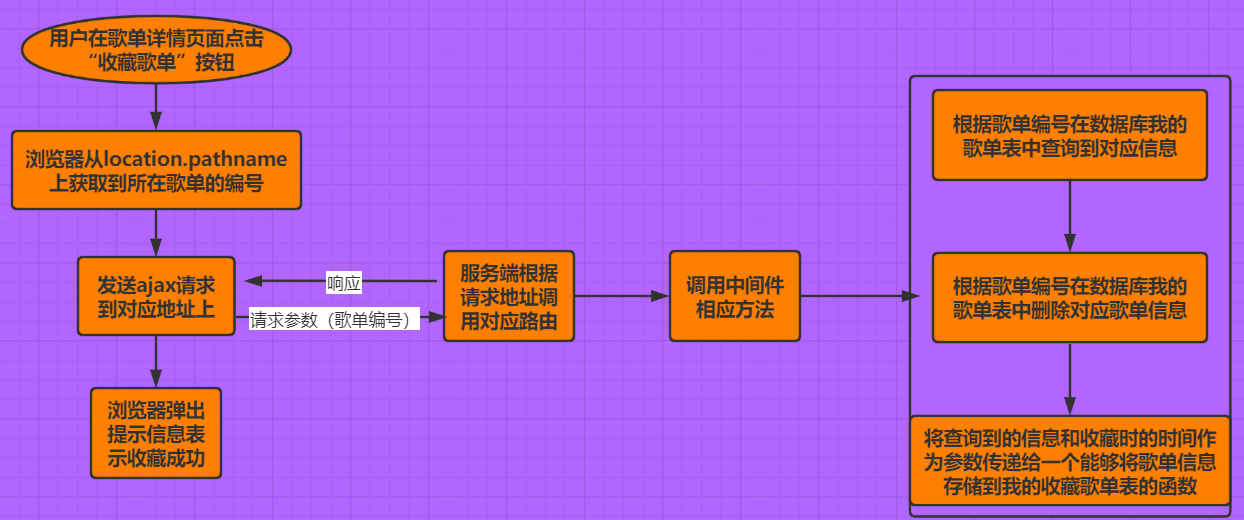
****

图 | 收藏歌单功能流程图

用户在个人中心点击我的收藏歌单板块后，浏览器需要渲染我的收藏歌单这一页面，服务端会提前在我的收藏歌单表中按照收藏时间的顺序将信息交给ejs模板引擎去渲染（渲染包括图片、歌单标题、播放量），渲染成功后用户就可以在我的收藏歌单板块中看到自己收藏过的歌单，在模板中还要给歌单封面和歌单标题的a标签指定对应歌单详情页的链接，这样就可以做到在点击我的收藏歌单后跳转到对应歌单的详情页面上去。

下面是制作收藏歌单功能模块流程图：

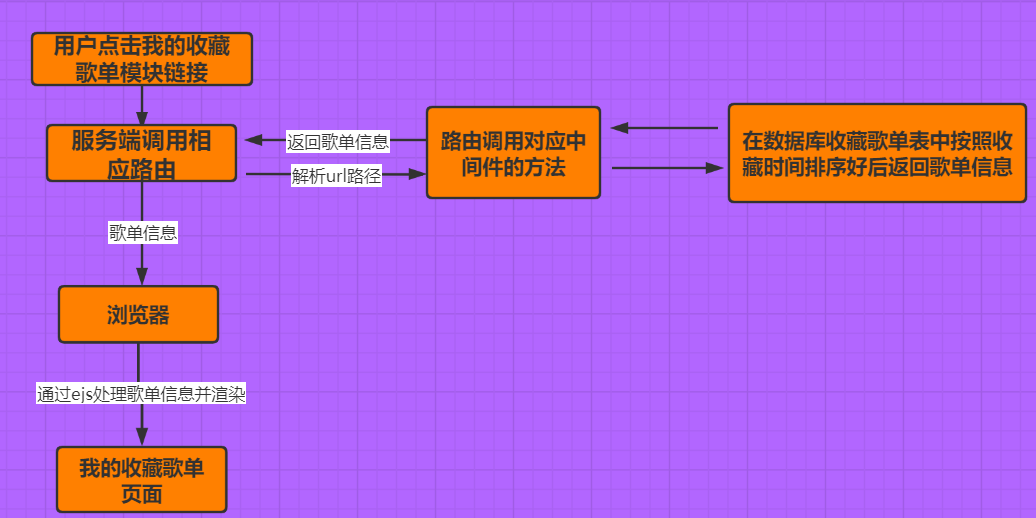


图 制作收藏歌单功能模块流程图

在我的收藏歌单详情页面中，我认为要给用户更多自己操作歌单的权利，于是我给该项目的我的收藏歌单模块添加了删除收藏歌单的功能，当用户点击收藏歌单之后，浏览器会获取其父盒子也就是装着单个歌单的li的innerhtml，再获取到整个收藏歌单的ul.innerhtml，通过split方法将前者在后者中删除掉后再次赋值给收藏歌单，这时用户就能够看到点击了“删除该歌单”连接后的歌单在我的收藏歌单中消失了。但是仅仅这样操作后，当页面一刷 新，页面又会恢复到之前的样子。要想真正地删除，还需要在数据库中我的收藏歌单表中删除掉才行。这步操作需要用到ajax 发送请求到负责删除歌单的地址上去，服务端会根据请求参数从数据库我的收藏歌单表中删除对应的歌单，这样刷新页面后仍然可以保持着之前的操作效果。

下面为在我的收藏歌单页面删除歌单功能操作流程图：

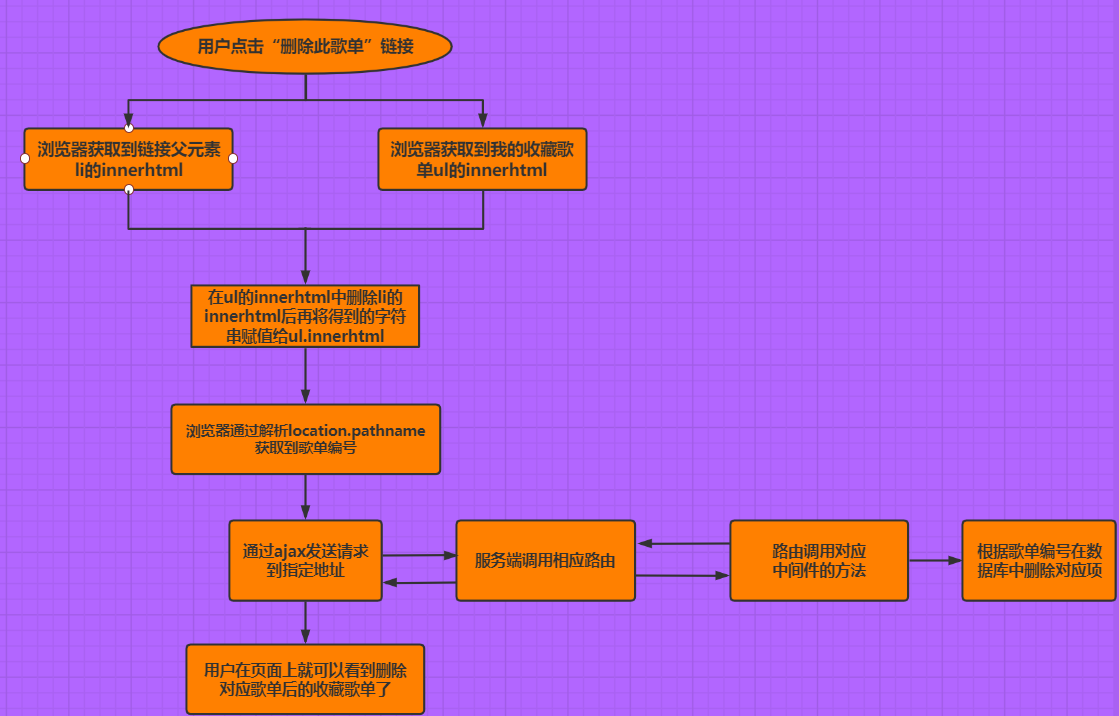


图 | 我的收藏歌单页面删除歌单功能操作流程图

**3.12 上传音乐**

为了能够让本项目成为开放性更强的平台，就要给用户更多的自主性和创造性，上传音乐功能就是一个很好的例子，用户可以将平台还没有的音乐或者是自己的音乐上传到平台中分享给其他用户，也是让本项目成为一个开放性网站的重要功能之一。

用户进入到上传音乐页面后，首先需要在下拉菜单中选择自己上传音乐所对应的分类，比如摇滚、流行、电子等等，选择完分类后点击文件框，在自己的文件夹中选择想要上传到平台的mp3格式文件，这里一次性只能选择一个，选择完成后，点击上传按钮，mp3音乐是通过表单提交到服务端的，而用户选择的歌曲分类是绑定了点击事件的上传按钮通过ajax发送请求到服务端的。

给浏览器发送ajax请求设置了定时器，要求在点击了上传按钮两秒后再发送请求，确保歌曲已经成功的上传到项目的mp3文件夹中。服务端对上传音乐的入口做了配置，确定好了上传音乐存放的目录，在接受到上传音乐的请求时，服务端调用中间件提前准备好的方法，此方法会通过解析重命名后的文件名来获取歌手名称和歌曲名称，并将此作为参数添加至我的上传音乐表中，随即还需要将此参数结合其他的默认参数传递到另一个函数当中，在数据库的全部歌曲表中增加新的一项。等到浏览器将用户选择的歌曲分类发送到服务端之后，服务端会调用相应的中间件，其方法首先执行的操作是获取列表中歌曲的总数，之后通过请求参数（用户上传歌曲对应的分类）在歌曲分类表中查找到对应的歌曲种类编号，在其then回调函数里面将歌曲的id值设置为歌曲总数，歌单名称就是请求参数，歌单编号是刚刚在各区分类表中查询到的数字，将这三个数据作为参数传递给一个能够在总歌曲表中更新数据的参数，调用完该方法后，浏览器会弹出提示信息表明上传操作已完成。这样操作后，总歌曲表中最新的歌曲数据就可以和用户选择的值一一对应了，上传歌曲的操作也算是最终完成了。

下面是上传音乐功能模块流程图：

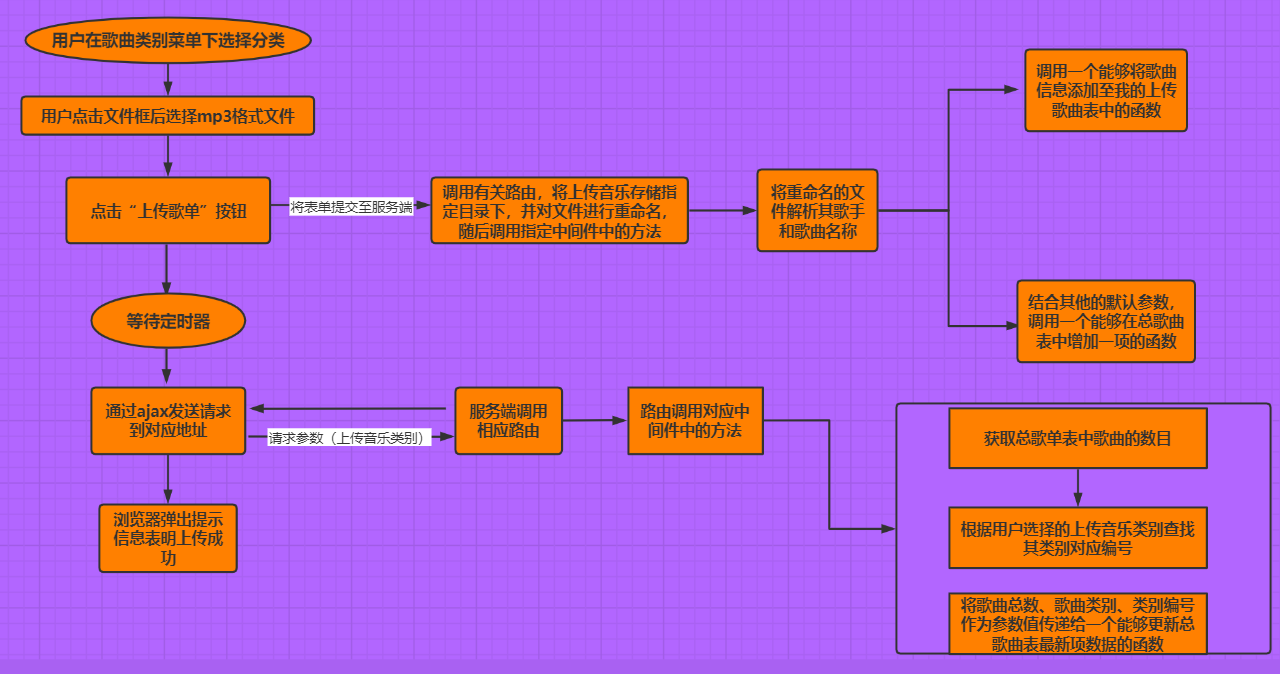


图 |上传音乐功能模块流程图

**3.13 制作歌单**

为了能让用户在平台中更加自由，可以展开想象，发挥自己的创新能力，在平台中添加了用户自主制作歌单的功能，使得每个用户都有制作歌单的权利，所有用户上传的歌单都会出现在最新歌单中和所有用户一起共享，极大程度上提高了用户与平台之间的互动，用户也可以明显的感受到自己也是平台的创作者之一。增加了用户体验。

**3.13.1 上传歌单标题**

用户在歌单标题文本框中输入完歌单标题之后，可以点击“上传标题”按钮进行上传。按钮绑定了点击事件，只要一被点击就会通过ajax往专门处理歌单标题的地址上发送请求，请求参数为歌单标题文本框里面的内容，服务单会调用对应的路由，随即路由又会调用相关中间件里的方法：首先在数据库我的歌单表中利用函数查询到歌单个数，随即获取当时的时间（也即上传音乐时的时间）、从req中获取到请求参数（歌单标题）、新建歌单的编号为歌单个数加一，将这三个数据结合默认参数一起参数传递给一个能够在我的歌单表中新建一项的函数，调用完该方法后，服务端会返回响应值，浏览器接收到会弹出“上传成功”字样的提示信息，这个时候通过歌单标题就已经在对应表中开辟了一项出来。

下面是上传歌单标题功能流程图：

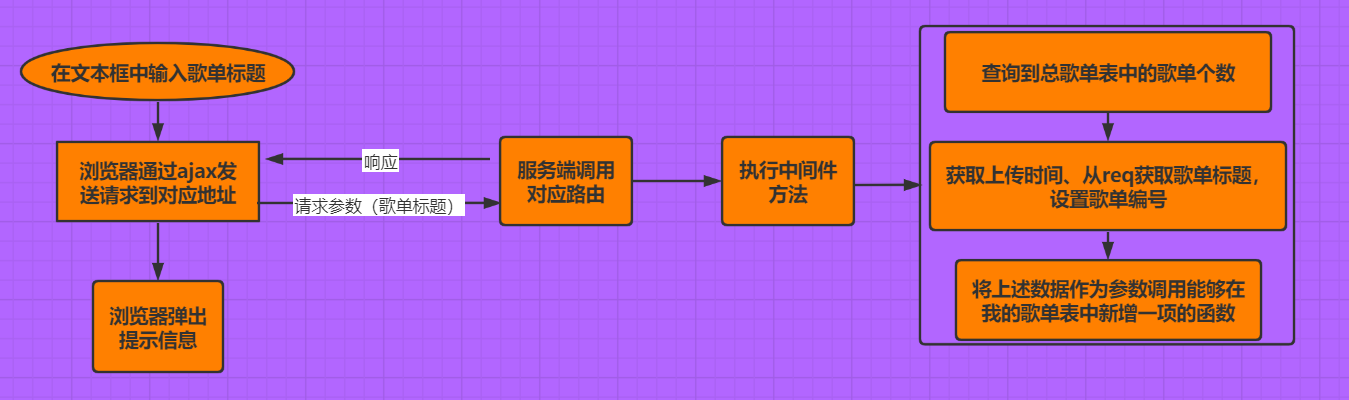
****

图 | 上传歌单标题流程图

**3.13.2 上传歌单封面图片**

用户在上传了标题之后可以在文件框中选择需要充当歌单封面的jpg或者png类型图片，选择完成后会通过表单提交到相应的地址，服务端会根据地址的不同调用不同的路由，路由会调用对应中间件的一个方法用来查询总歌单表中歌单的数目并返回信息，服务端也早已对上传图片进行了配置即上传的图片会被放到指定的目录下面，服务端还要通过文件的原始文件名结合刚刚查询到的歌单数目给上传的图片文件重新命名，此文件中的数字即为其对应歌单的编号，所以下次再渲染歌单的时候就能够一一对应上，该方法执行完毕之后，服务端会返回响应信息，由于使用的是表单提交，所以提示信息会在一个空白的页面中显示，用户看到提示信息之后可以点击浏览器的返回键往下在制作的歌单中上传歌曲。

下面是上传歌单封面功能流程图：

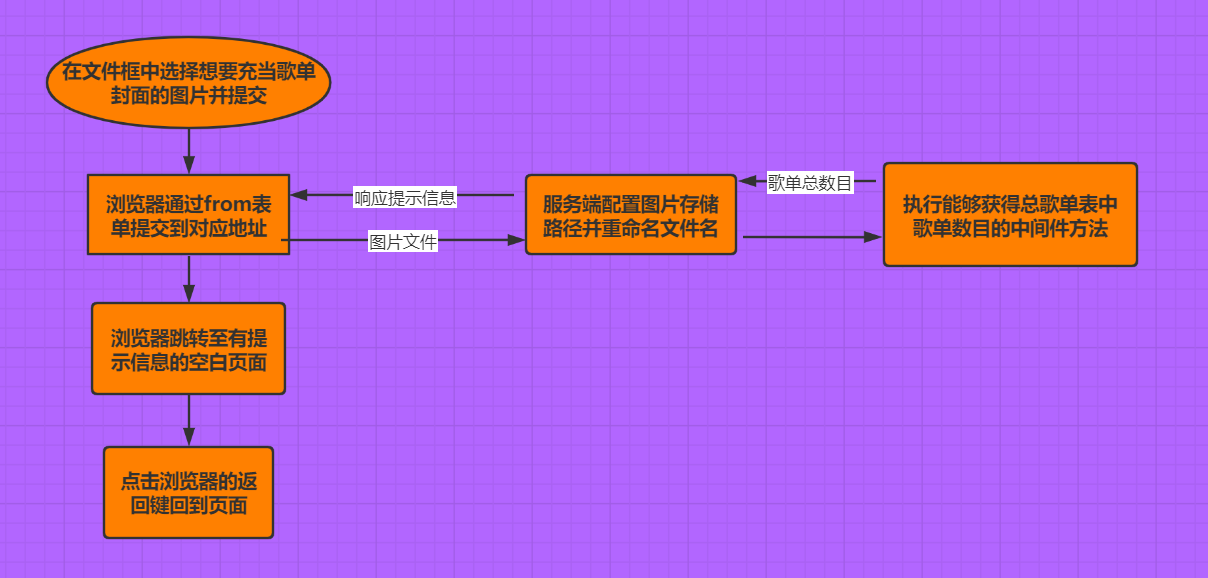
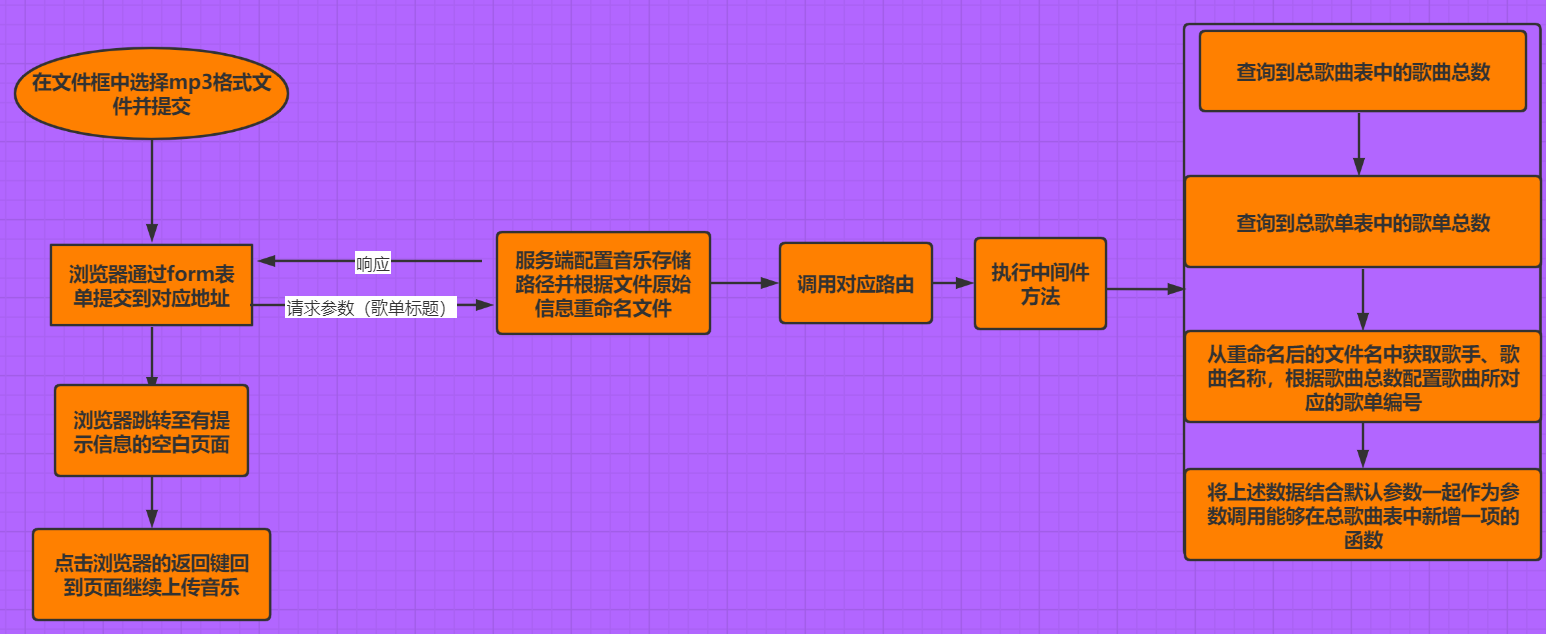
****

图 |上传歌单封面功能流程图：

**3.13.3 往歌单中上传歌曲**

跟上传音乐功能有很多相同之处，都需要在文件框中选择单个mp3格式文件后通过点击按钮进行上传，上传的音乐会在服务端的配置下存放到对应的目录下，还是通过先获取到总歌曲表中的歌曲数目，然后通过结合并解析文件的原始名称进行重命名新上传的音乐文件。重命名完成后服务端会调用对应的路由用来匹配上传的歌曲与新建歌单，仍然是通过调用中间件方法的形式完成此操作，在该方法中，先执行获取总歌曲表歌曲总数的函数，其会返回查询到的数目，该数据的获取是为了上传歌曲编号的确定，其次，还会执行获取总歌单表中歌单总数目的函数，利用该数据是为了确定新上传歌单的编号，有了这些之后，该方法通过解析重命名文件的名称获取到歌手和歌曲名称，将上传歌曲的编号设置为总歌曲数目加一，将获取到的歌手名称、歌曲名称、歌单编号结合默认参数，一起作为参数传递给能够往总歌单表中开辟新的一项的函数，这样操作过后，上传的歌曲再数据库中就与新建的歌单一一对应了，方法执行完毕后，服务单会响应提示信息用于提醒用户歌曲已经成功添加至歌单，由于请求是通过表单提交发送的，所以响应的提示信息会在空白的页面中呈现给用户。

下面是往新建歌单中上传歌曲功能流程图：

图 |往新建歌单中上传歌曲功能流程图

**3.14 自定义歌曲分类**

用户在个人中心板块通过点击“自定义歌曲分类”可以进入对应页面，用户可以在输入框内输入想要自定义的歌曲类名，输入完成后点击提交按钮即可，在按钮被点击后，浏览器会通过ajax发送请求到指定地址上去，请求参数是文本框中的值（即用户想要自定义的歌曲类别名称），服务端会在相应路由里调用相关中间件方法该方法里面会依次执行两个函数，第一个函数是为了获取数据库歌曲种类表中所有类别的数目，目的是为了给新自定义的分类设置编号，该编号就是歌曲类别总数加一，其次，还要获取到当下的时间（就是记录下歌曲分类的创建时间），通过req还可以查询到ajax的请求参数（自定义分类），这样服务端就有了三个数据，分别是自定义歌曲分类名称和其编号以及创建分类时对应下的时间，可以将这三个数据作为参数传递给第二个函数，这个函数的主要作用就是在数据库歌曲分类表中根据各参数开辟新的一项，此方法调用完后，服务端会发送响应值表示操作结束，浏览器接受到响应后会弹出提示框表示“歌曲分类已被成功添加”。

该分类被创建后，用户可以到上传音乐模块，里面的选择歌曲分类已经有了自己刚刚定义的那一个分类，用户可以为刚刚定义的分类上传歌曲。定义分类结束后，用户可以通过回到首页面，在最新歌单旁边的歌曲分类中，可以看到自己新定义的歌曲分类，也可以通过点击“更多”链接跳转至全部歌曲分类详情页面上去，里面也能够看到最新的那个分类，通过点击该分类，可以跳转至对应歌曲分类的详情页面。

下面是自定义歌曲分类功能流程图：

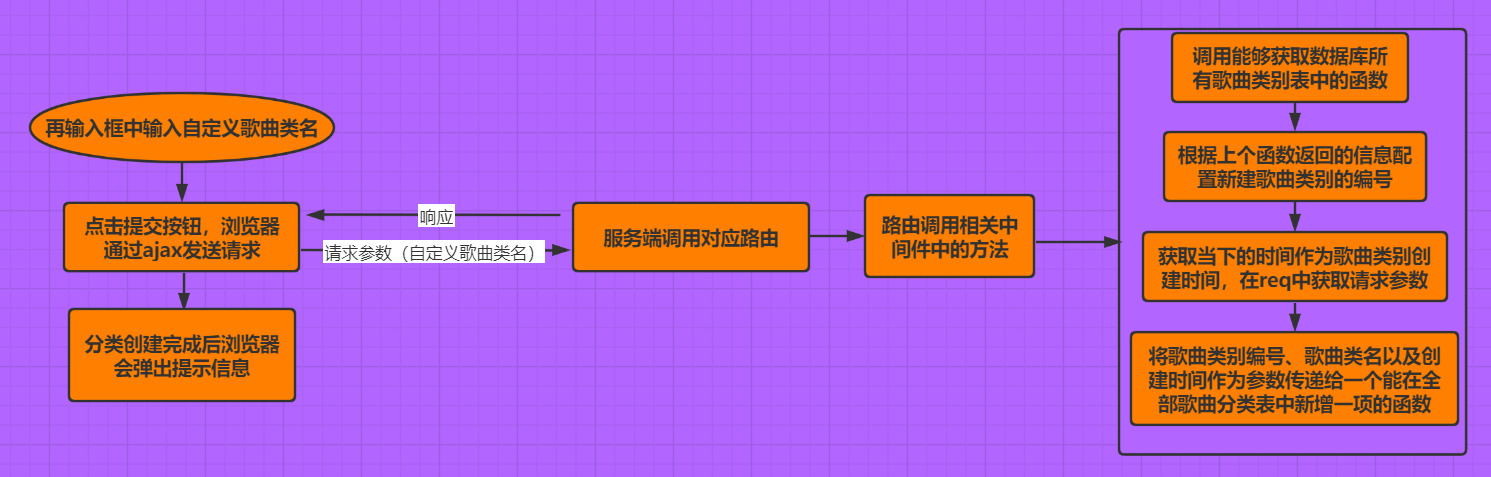


图 | 自定义歌曲分类功能流程图

**3.15 将歌曲添加至播放列表**

为了使得本项目的功能更加丰富以及多元化，能更好的面向用户，所以自己在考核功能之外添加了一些对提升用户体验有帮助的功能。在这期间我参考了各大流行音乐平台所具有的功能，发现其中有很多可以方便用户使用的模块，也能够使得该音乐网站更加人性化和贴近用户，“将歌曲添加至播放列表”就是其中一个我认为不错的功能，所以也是通过自己的方式添加到了本项目中。此功能与“点击按钮播放音乐”功能有相同之处，都会将音乐添加至播放列表中，但是前者会直接播放音乐并将该音乐添加到播放历史记录中，而后者则是将歌曲添加至播放列表后不直接播放该音乐，也不会将歌曲添加至播放历史记录中，而是等用户执行操作，用户可以通过在播放列表中“点击其对应播放按钮”来播放该音乐并添加至历史记录中去。这一项功能在歌单详情页中、歌曲类别详情页中、搜索歌曲详情页面、我的收藏歌曲页面以及我的播放历史页面中都存在，几乎有歌曲的地方都可以执行此操作，普及性很强。

在用户点击了“添加至播放列表”这一按钮后，浏览器会先获取到该按钮的属性id值（也就对应歌曲的编号），这是服务器用模板引擎渲染出来的，随即根据ajax发送请求到相应地址，服务端根据不同地址调用不同路由，路由又会调用不同的中间件方法，该方法首先会先获取请求参数（要添加值播放列表的歌曲编号），将该编号作为参数传递个一个能够在数据库播放列表中删除对应歌曲的函数，这一操作目的是为了避免用户操作失误导致重复添加了同一首歌曲到播放列表，不利于用户体验。执行完删除操作之后，还需要将歌曲编号作为参数传递给一个能根据歌曲编号在总歌曲表中查询到对应歌曲信息的函数，在这个函数的then回调中，需要利用查询到的信息解析出歌曲编号、歌曲名称、歌手名称等信息，然后获取到当前的时间，上述信息结合起来一起作为参数传递给一个能够在数据库全部歌曲表中新增一项的函数，查询到的信息会放到req中，目的是为了等等在路由里能够将查询信息作为响应值发送到前端，这一系类操作结束后，在数据库对应表中已经成功添加了该歌曲，但是在页面中的播放列表中还没有显示出来，所以还需要在ajax里的success函数通过响应值解析出对应歌曲的歌手以及歌曲名称，通过字符串的形式将其对应的li拼接好，获取到播放列表ul的innerhtml，先利用replace在ul上将两者相同部分清除，然后再将拼接好的字符串与删除后的ul进行拼接，形成新的字符串后赋值给ul.innerhtml并将播放列表显示出来，这样用户就在播放列表的第一项看到了自己刚刚添加的歌曲，由于已经存到了数据库中，所以即使刷新了页面过后，添加的歌曲仍然存在播放列表中。

下面是将歌曲添加至播放列表功能流程图：

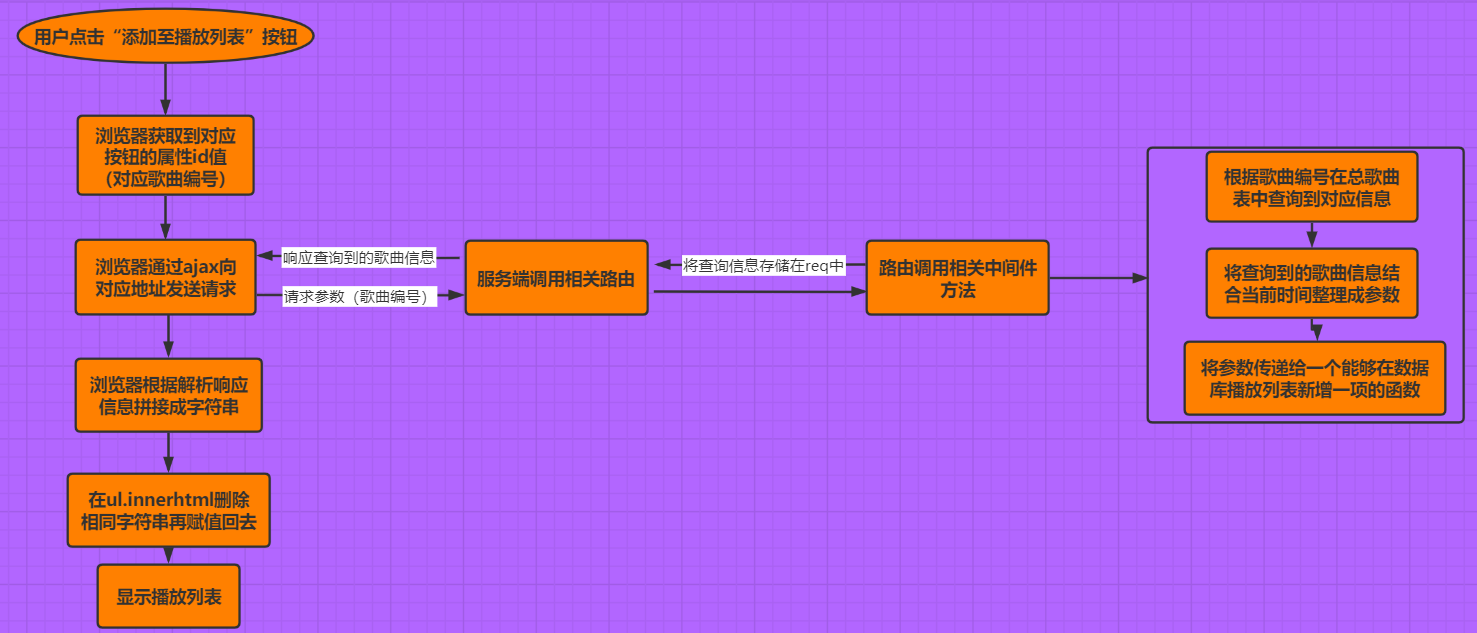


图 | 将歌曲添加至播放列表功能流程图

**3.16 收藏播放列表全部歌曲**

其实在上文中已经展示过了收藏单个歌曲功能，但如果遇到了用户想要收藏播放列表的所有歌曲或者是歌单里面的所有歌曲时，需要自己一首一首歌曲去添加，要是遇到了很多首歌曲需要添加的情况，对于用户来说其实是非常耗时的，所以在本项目的播放列表中，为了增加用户在改平台的体验，给用户增加了收藏全部歌曲的功能。

用户在播放列表中点击了“收藏全部”链接之后，如果播放列表为空，则浏览器会弹出警示框提示用户播放列表中没有歌曲。只要播放列表不为空，浏览器就会通过ajax 向相应地址发送请求，服务端对应路由接收到请求后，会调用有关的中间件，在中间件的方法中，首先调用了一个能够查询数据库总歌单表中所有信息的函数，在此函数的回调函数中，根据该函数的返回值（是一个数组），利用foreach循环遍历，数组中的每一个对象都会结合当前的时间经过解析成对应参数后，调用能够在我的收藏歌曲表中新增一项的函数，这样播放列表里面所有的歌曲都会按照顺序依次添加至我的收藏歌曲表中，为了避免用户此次要添加的歌曲已经存在于收藏歌曲中，在把歌曲添加至播放列表之前，先执行了在我的收藏歌曲表中删除对应歌曲的操作，这样新收藏的歌曲就会覆盖掉之前收藏的歌曲，用户在下次跳转至我的收藏歌单页面后，就能够看到播放列表所有的歌曲出现在我的收藏页面当中了。

下面为播放列表中收藏全部歌曲功能流程图：

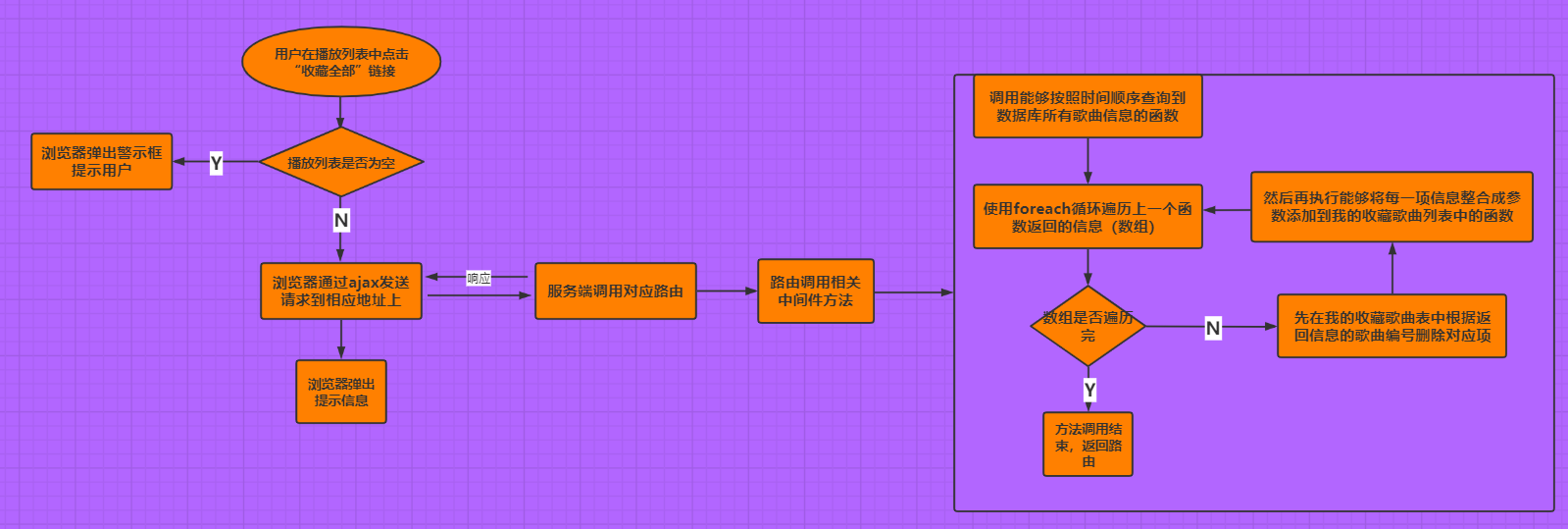


图 |播放列表中收藏全部歌曲功能流程图

**3.17 在播放列表或收藏歌曲中删除歌曲**

每个页面渲染歌曲模块的时候，都会给对应标签的id属性值赋予相应的歌曲编号，如此操作，每一个“删除”链接的id值即为其对应歌曲的编号。以在播放列表中删除歌曲为例，每当用户点击了“删除”链接之后，浏览器会先获取到该链接的属性id值，然后通过ajax发送请求到对应地址上，请求参数为链接的id值，服务端根据请求地址调用对应的路由，然后路由在调用相关的中间件方法，在该方法中，首先会在req中查询到请求参数（也就是歌曲编号），然后将该歌曲编号作为参数传递给一个能够根据参数在数据库我的播放列表中查找对应歌曲的函数，该函数执行完毕后，会先将查询到的结果信息存储到req中，便于路由响应信息，然后在此函数的then回调中调用一个能够根据歌曲编号在播放列表中删除对应歌曲的函数，此方法执行完之后，服务端会将查询到的歌曲信息返回给前端，浏览器根据信息解析出歌手和歌曲名称，并嵌套在模板字符串中，然后浏览器获取到播放列表ul的innerhtml，使用replace与模板字符串相比较，删除掉一样的部分后再渲染到将此字符串赋值给ul.innerhtml，从而将删除歌曲后的播放列表信息渲染到页面当中去。

在我的收藏列表中删除歌曲跟在播放列表中原理一致，这里不做过多赘述。

下面为在播放列表中删除歌曲功能模块流程图：

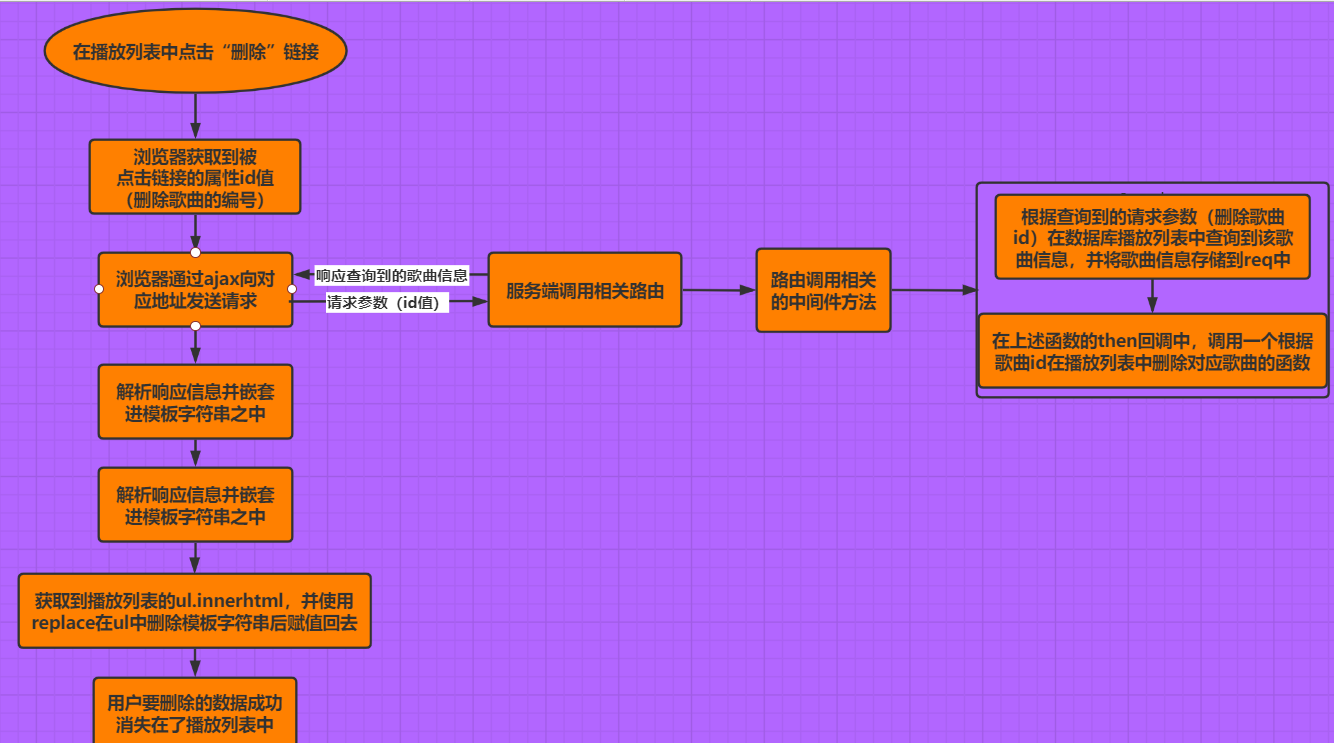


图 | 在播放列表中删除歌曲功能模块流程图

**3.18 进入歌单使其热度加一，播放歌单使其播放量加一**

由于首页上的热门歌单推荐是根据数据库我的歌单表中按照热度排序所查询出的数据，如果不能时间歌单热度的变化，则无论用户怎么操作，热门推荐的歌单永远都只会是固定的，但这样限制了平台的多样性和更多的可能性，所以站在开发者的角度上考虑，每次进入歌单之后，都会使得该歌单热度加一，当某个歌单在一段时间内比较火热，点击比较频繁的话，那么该歌单很有可能就会在下一次刷新时排到其他歌单前面去，获取原本不在热门推荐歌单中的歌单，由于热度排名靠前也会被选如热门歌单的行列，这样不仅使得整个项目平台更加多元化，也充斥了更多的可能性，同时也能股作为一种动力激发创作者制作出更加优秀、受众更广的歌单。播放歌单的制作目的与热度相似，用户每在相应的歌单中点击“播放全部”按钮，歌单详情页中的播放量就会加一，可以加深用户对平台真实性和准确性的认可程度。

每次在首页点击歌单封面或者歌单标题后，服务端都能根据其对应a链接的地址来接收请求，然后从各种中间件中获取到相关信息后再利用模板引擎渲染到对应歌单详情页面上去。为了能够实现进入歌单详情页面热度就加一的功能，需要在原先的路由再添加一个中间件并执行其对应的方法，该方法具体步骤通过req.params查询到对应歌单的编号，然后调用一个能够根据歌单编号查询到相应歌单信息的函数，并把查询信息存储到req中方便响应回前端，此后在此函数的then回调中，能够获取到该歌单原先的热度，将该热度加一后结合歌单编号作为参数传递给一个能够根据歌单编号更新其热度的函数，这样数据库的更改就已完成，下次再进入歌单详情页面的时候，其呈现出的热度就会比上次查看时增加1。

在歌单详情页面中，实现点击“播放全部”按钮后播放量加一的操作其实和进入歌单后使其热度加一的操作有很多相似之处。当用户点击了“播放全部”按钮后，浏览器会先利用location.pathname获取到对应的歌单编号，随后再通过ajax向对应地址发送请求，请求参数就是刚刚获取到的歌单编号，服务端会根据相应的地址调用路由，该路由会调用相关的中间件中的方法，在该方法的前一段代码是在为点击“播放全部”的另一半功能——也就是自动播放第一首歌曲并添加至历史记录和将歌单中的所有歌曲添加至播放列表所准备的，下段代码才是对应这个功能的，首先要执行一个能够根据歌单编号在总歌单表中获取到对应的歌单信息的函数，然后在此函数的then回调中，获取到该歌单的播放量，根据该播放量配置参数，参数的值就是播放量加一，配置好后结合歌单编号调用一个能够在根据歌单编号在总歌单表中对应歌单项更新播放量数据，该方法调用完后响应歌单信息到前端，前端先进行判断播放量有无过万，若没有过万，则会直接在页面上使播放量数值加一。

下面为点击“播放全部”按钮播放量加功能流程图：

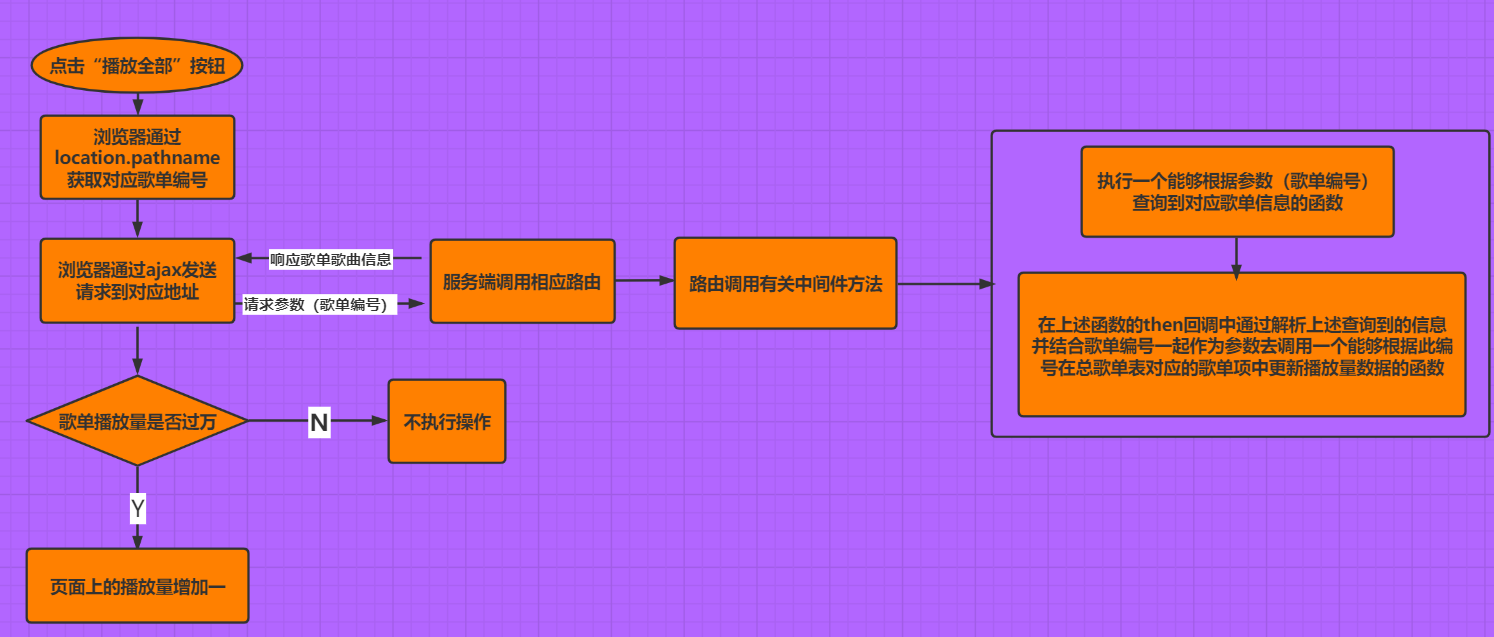
****

图 | 点击“播放全部”按钮播放量加功能流程图