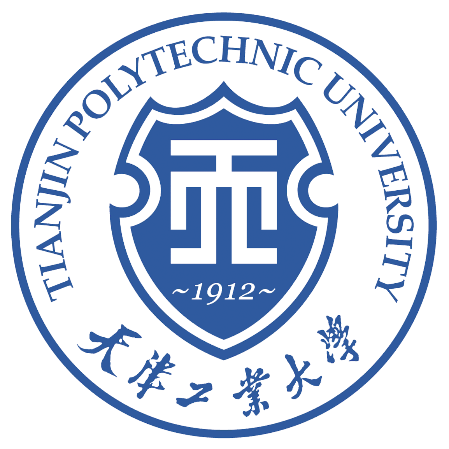
天津工业大学计算机科学与技术学院

实训总结报告



　项目名称: 基于豆瓣资源的个性化信息整合库平台

　合作培养企业：天津市融创软通科技股份有限公司

　专业方向：大数据方向

　项目负责人：何晶

　专业班级: 软件z1801

姓名：樊午龙

　实训起止日期：2020.6.15-2020.7.15

**1 引言** （一级标题，小三黑体，居中）

1.1项目概述（二级标题，四号黑体，左对齐）

说明项目目的、背景、意义及参考资料（包括企业提供的素材、DEMO）等；

随着科技时代的发展，我们的社会也在飞速的发展。作为一个信息化的社会，我们的衣食住行都离不开各种各样的信息。例如我们习惯网购各式各样的衣服，在网上搜索各种各样的美食，然后网站收集你习惯搜索的信息，再为你进行个性化推荐。毫不夸张的说，我们的生活已经离不开各种各样的信息了，我们每一个人每时每刻都在产生大量的数据。而就是为了处理如此海量的数据，大数据这门研究就应运而生了。

我们这个项目叫做基于豆瓣资源的个性化信息整合库平台，其目的就是做一个可以收集大量数据将其插入数据库，并在后台提取出来对数据进行分析整合的平台。这么做的意义在于我们可以收集大量我们想要的数据，然后针对我们想要的方面进行统计分析。例如拿我所爬取的电视剧数据来说，我们可以统计哪一年评分最高的前X名的电影，亦或是可以统计5星级评分的整体分布，甚至可以精确到某一条评论来估计是正面情绪还是负面情绪。我们将结果以图表的形式展现，这样可以更直观的得到我们想要的结果。通过一系列功能的实现，用户可以更好地检索自己想要的东西，我们也可以通过海量数据更好地了解用户，更好地为用户服务。

参考资料：通过老师对各个模块的讲解，以及老师写的简单例子，加上网上资料的自学，逐步搭建起整个框架。

1.2 需求分析

为什么做这个题目，有什么实际价值

我们身处于信息化的时代，不仅我们想要的东西可以快速的检索出来，我们自己无时无刻也在产生大量数据，这就是大数据给我们生活所带来的实际影响。但是杂乱无章的数据是没有任何意义可言的，我们也无法快速找到我们想要的数据，更别提统计了。这就需要我们利用技术对数据加以整合分析，留下我们想要的数据，剔除我们不要的数据，然后将其规范化，最后可以利用得到的数据进行我们想要的分析，将结果进行可视化，这是我们做此项目的初衷。做出这个项目我们可以从任何一个网站使用爬虫爬出我们想要的数据，将其用一张规范的二维表整合进数据库，然后使用BS项目的开发结合前端的开发，将得到的数据提取出来，用前端技术做成一个平台，我们可以在这个平台上检索我们想要的数据，亦或是可以选择一些项目进行分析，用eCharts技术进行可视化，使展示更加直观、美观。这就是此项目的意义。

同时我们在此平台加入用户管理功能，使用此平台前需要登录，如果没有账户则需要注册。此外用户还分为普通用户和管理员用户，二者的权限有所差别，同时管理员用户可以管理普通用户。并且用户拥有标记自己喜欢的或者已看过的图书、电影的功能。

1.3 运行环境

软、硬件环境

（正文，小四号宋体，1.25倍行距）

软件环境：Gitee负责版本管理，提交项目；Pycharm负责搭建scrapy框架，利用爬虫爬取数据；IDEA负责服务器端以及前端代码编写——服务器端开发框架SSM（Spring+SpringMVC+MyBatis），前端开发技术（jQuery+LayUI+eCharts）；连接远程数据库MySQL(Mariadb)；平台开发模式：基于JSON数据的前后端分离式开发。最后使用浏览器打开最终成型的平台。

硬件环境：运行内存8G的PC即可。

**2 项目设计**（小三黑体，居中）（以个人主要负责内容为主）

2.1 设计思路

个人主要负责音乐模块的数据爬取、数据存储与数据分析，设计思路如下：

1. 使用python的scrapy框架，使用pycharm作为项目开发工具，对豆瓣的音乐数据进行爬取。
2. 爬取思路：以音乐标签页为起点，[www.douban.com](http://www.douban.com)为边界，对每个标签依次爬取。每个标签内为翻页栏目式，将页面内每首音乐依次爬取其详情页url后通过翻页按钮进入下一页，以此循环爬取。在每个音乐详情页内爬取所需数据后，再爬取对应评论信息，并爬取评论用户的详情页链接。在用户详情页内爬取所需数据。
3. 代码设计思路：将爬取内容分解为四种样式，标签、栏目、音乐详情、用户详情、根据所爬不同的页面进行不同的数据提取(爬取)[music.py内涉及]，根据不同的样式调用不同的数据库存储方式[piplines.py内涉及]。
4. 使用mySQLworkbench对已爬取数据进行初步处理。对数据单元内多余的单引号进行清除，对缺少关键数据的条目进行删除等。
5. 使用java BS架构中的mybatis框架，对前后端页面进行开发。功能模块大致分为五类，书籍、音乐、电影、电视剧、用户管理。

2.2 模块功能介绍

音乐模块功能介绍：

音乐模块对豆瓣的音乐数据进行采集、存储、分析、展示、让用户可以一目了然的了解豆瓣音乐的一些突出特征。

首先，音乐模块支持乐曲信息查询。查询项包括乐曲名、歌手名、音乐评分。结果项包括豆瓣网音乐专属ID，音乐名、表演者、音乐又名、音乐流派、专辑类型、音乐介质、发行时间、音乐评分、评价人数、五星占比、四星占比、三星占比、二星占比、一星占比、音乐链接。且支持模糊查询，即使记忆不清，也可以找到用户想要的音乐。

其次音乐模块对给予五星好评最多的用户的居住城市进行排行，以及各评分音乐数进行统计并展示。

最后对音乐模块的使用权限进行限定，查询功能完全开放，数据分析的查看功能则需要管理员权限。

2.3 模块结构图

**前端页面数据**

豆瓣书籍

豆瓣音乐

豆瓣电影

豆瓣电视剧

用户信息

**豆瓣网数据**

豆瓣书籍

豆瓣音乐

豆瓣电影

豆瓣电视剧

**数据库数据**

豆瓣书籍

豆瓣音乐

豆瓣电影

豆瓣电视剧

用户信息

书籍模块支流

爬虫1、2、3、4分别爬取

用户模块支流

用户模块支流

电视剧模块支流

电影模块支流

音乐模块支流

音乐名称

音

乐

数

据

作者名称

乐曲评分

各评分段数量

五星好评城市来源榜

2.4 程序流程图

2.5 功能设计分工

需详细标注每部分的内容，具体任务及任务安排等。

**3 详细设计**（小三黑体，居中）

详细写明项目的设计（系统设计、数据库设计）等，并写明对所涉技术、核心功能等描述；

系统设计：

PYTHON部分：

使用scrapy框架，都豆瓣音乐数据进行按类爬取。共分为四类，标签类，栏目类，音乐详情类，用户详情类。针对每一类都从item到piplines进行单独规划设计，目的在于针对特定页面进行特定搜索，以此减少不必要的搜索，提高爬取效率。TAG类：由于tag页面只有搜索链接，没有需要存储的数据，所以在只需遍历并压入待爬取队列即可，piplines中为空。TABLE类：此页面需要我们1.存储每一个栏目的音乐id以及详情页面的链接，在数据库中建立条目。2.将详情页链接压入待爬取队列中。3.将指向下一页的翻页链接压入待爬取队列中。

DETAIL类：1.此页面需要爬取大量数据，按音乐id查找并存入数据库中。2.我们还需要存储最多三条用户评论信息，并按照以该音乐id以及用户名为主键在评论数据库添加条目。此外，每次录入前需要验证数据库内是否已有该条目，若存在则执行更新操作，若不存在则执行插入操作。

USER类：此页面我们只需要读取一些特定用户信息，并以用户名为主键在数据库内执行更新操作即可。

JAVA部分：

使用mybatis框架，将页面数据通过js->controller->service->mapper到层，通过mapper层的底层实现达成前端页面与数据库的交互操作

前端页面包括用户模块(登录、注册、检索)，主体模块(书籍、音乐、电影、电视剧)，主题模块内每一分类又分为多个模块，包括查询模块以及多个不同的数据分析展示模块。

数据库设计：

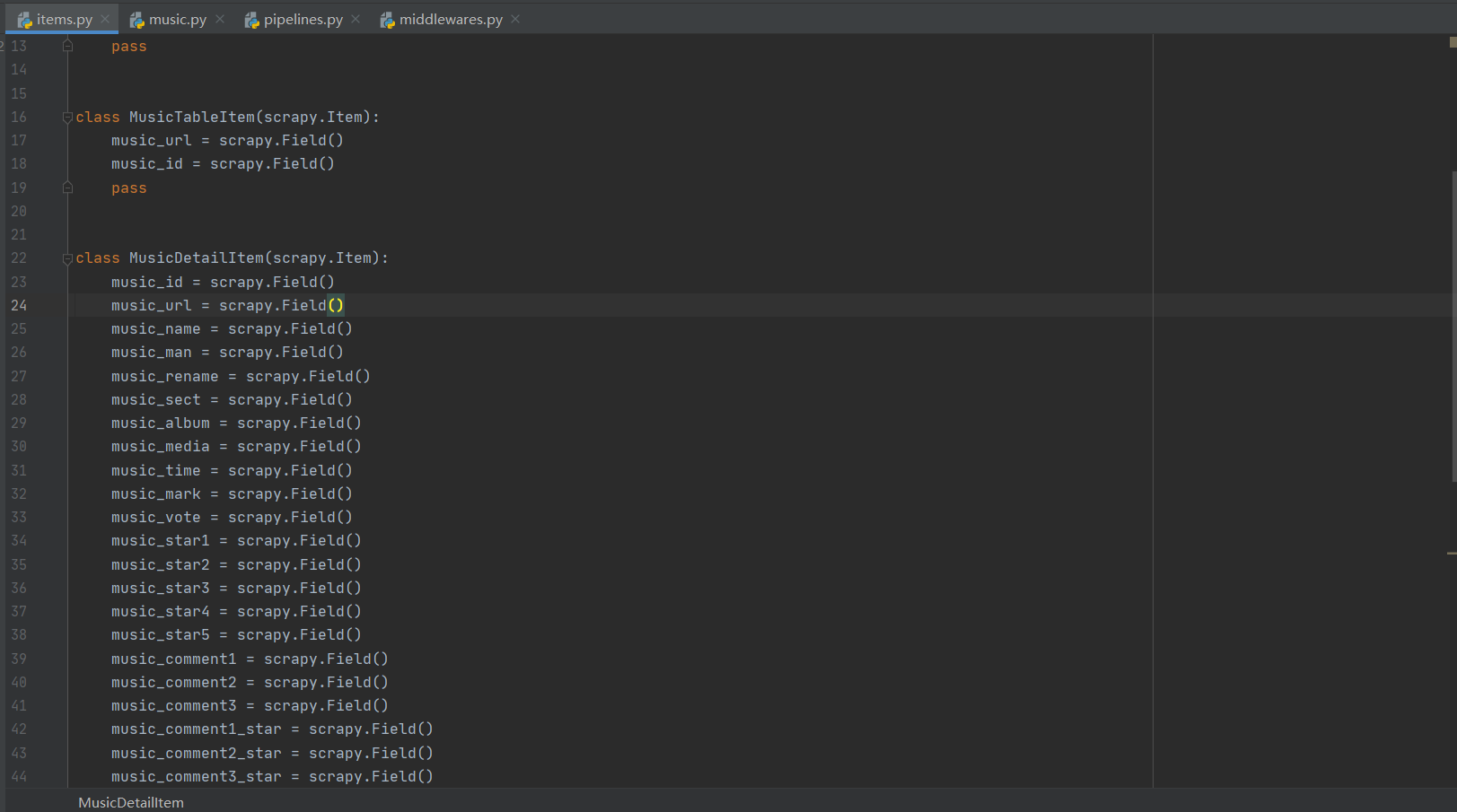
数据库建立三张表，分别为音乐条目表，评论条目表，用户管理表。音乐条目表以音乐id为主键，包含其余所有音乐信息。评论条目表以音乐id与评论者用户名为主键，包含评论内容、评价星级、评论者常住地、评论者创号时间。用户管理表包含用户名、密码、昵称、用户状态、用户头像存储路径。

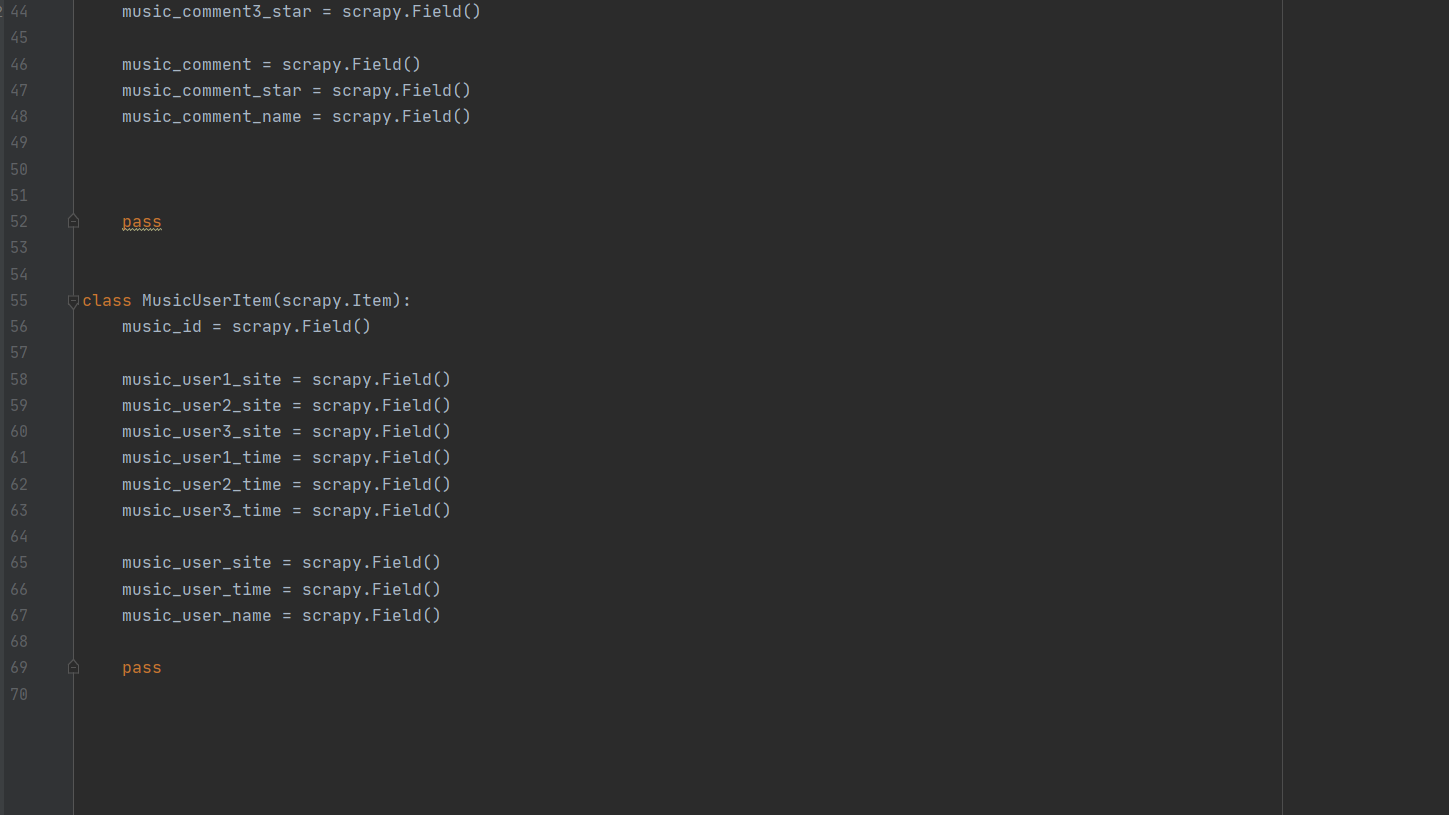
**4 实现方案**（小三黑体，居中）。

详细写明项目功能实现描述、核心代码及实现结果等；

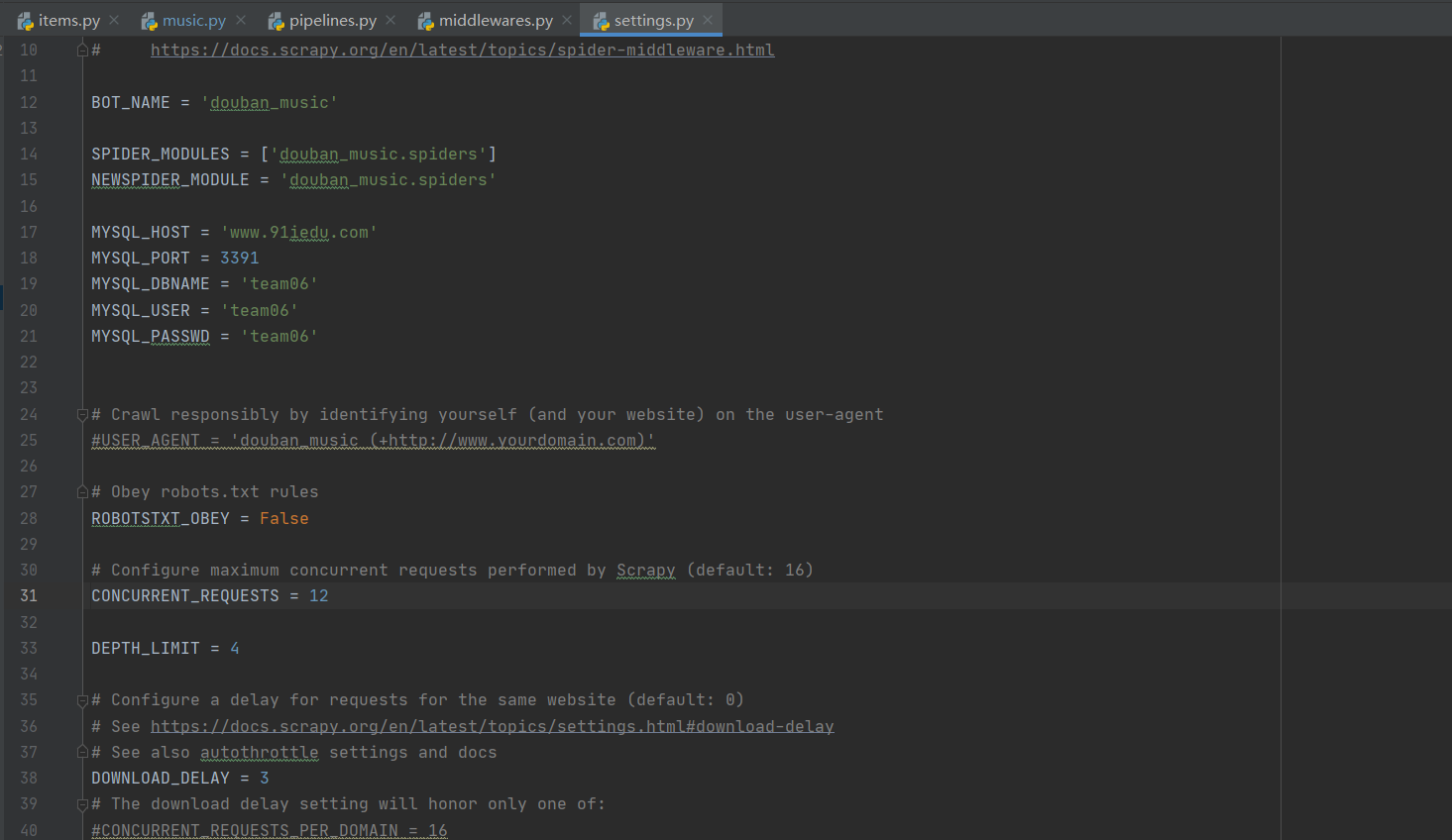
并对调试好的程序，从执行程序弹出界面开始，每一步操作截一个图，并附加说明（图片格式为JPG格式）。

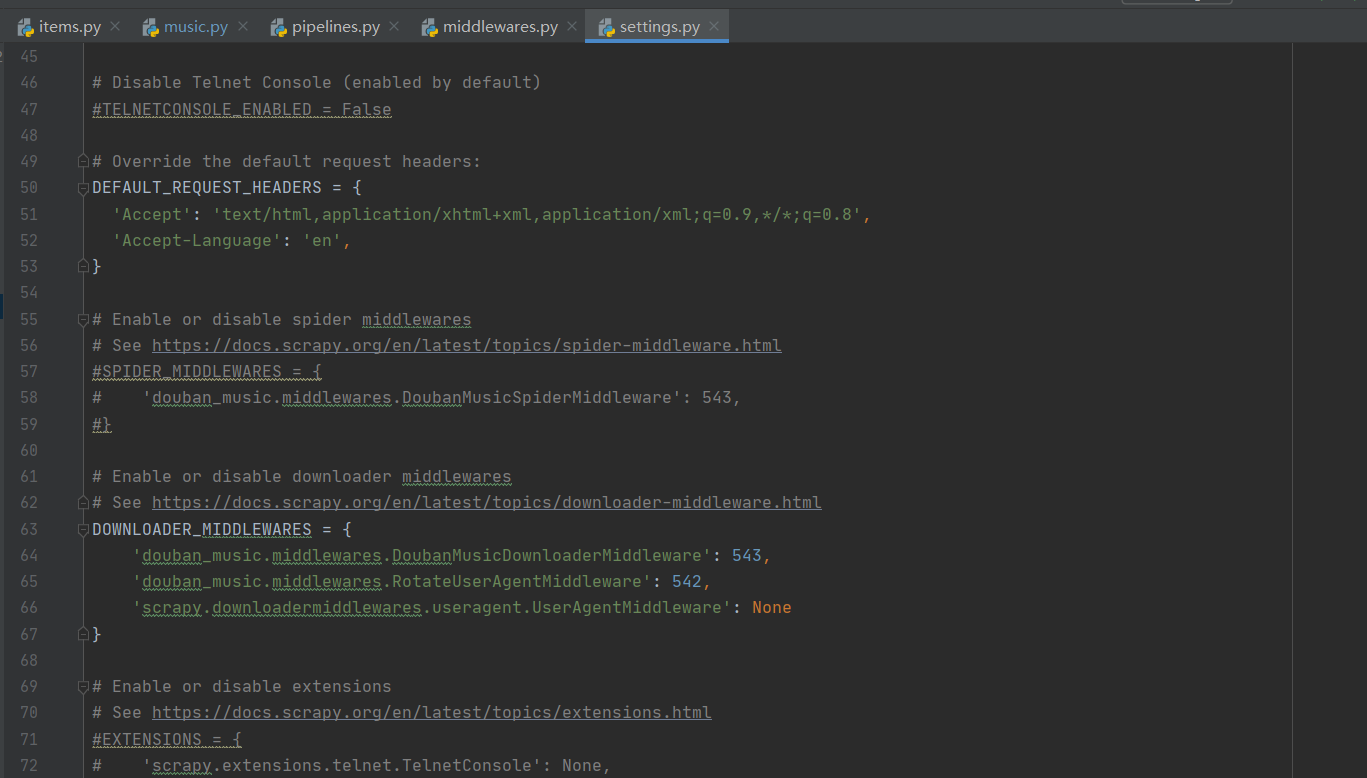
首先从数据爬取开始。



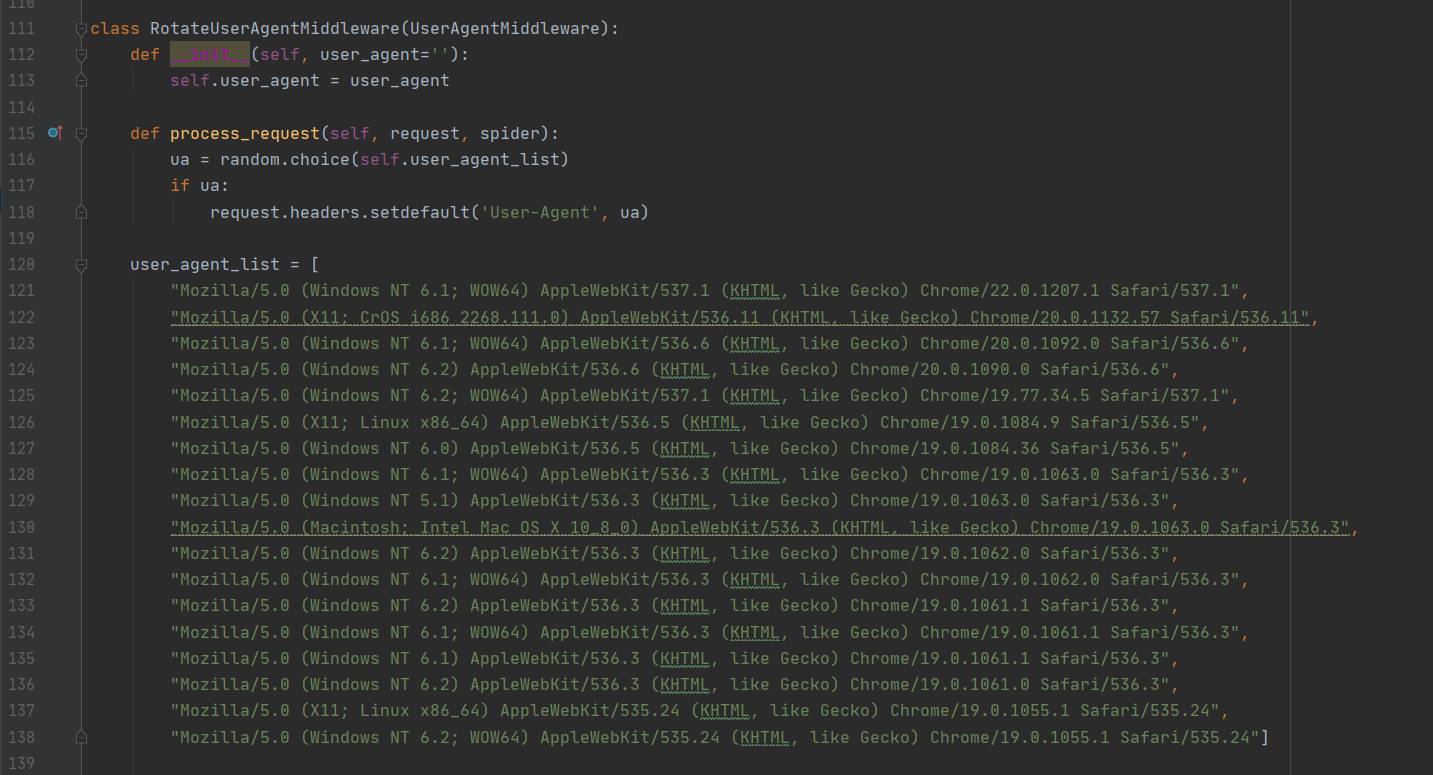


最基础的，定义所有要用到的变量，根据爬取页面不同会调用不同的item，所以在各自item中定义当前页面所需变量。





其次，在setting中我们也要进行一些配置，比如定义数据库链接的基本常量、开启middlewares等



这里我们设置了浏览器访问池，减少被封禁的风险。



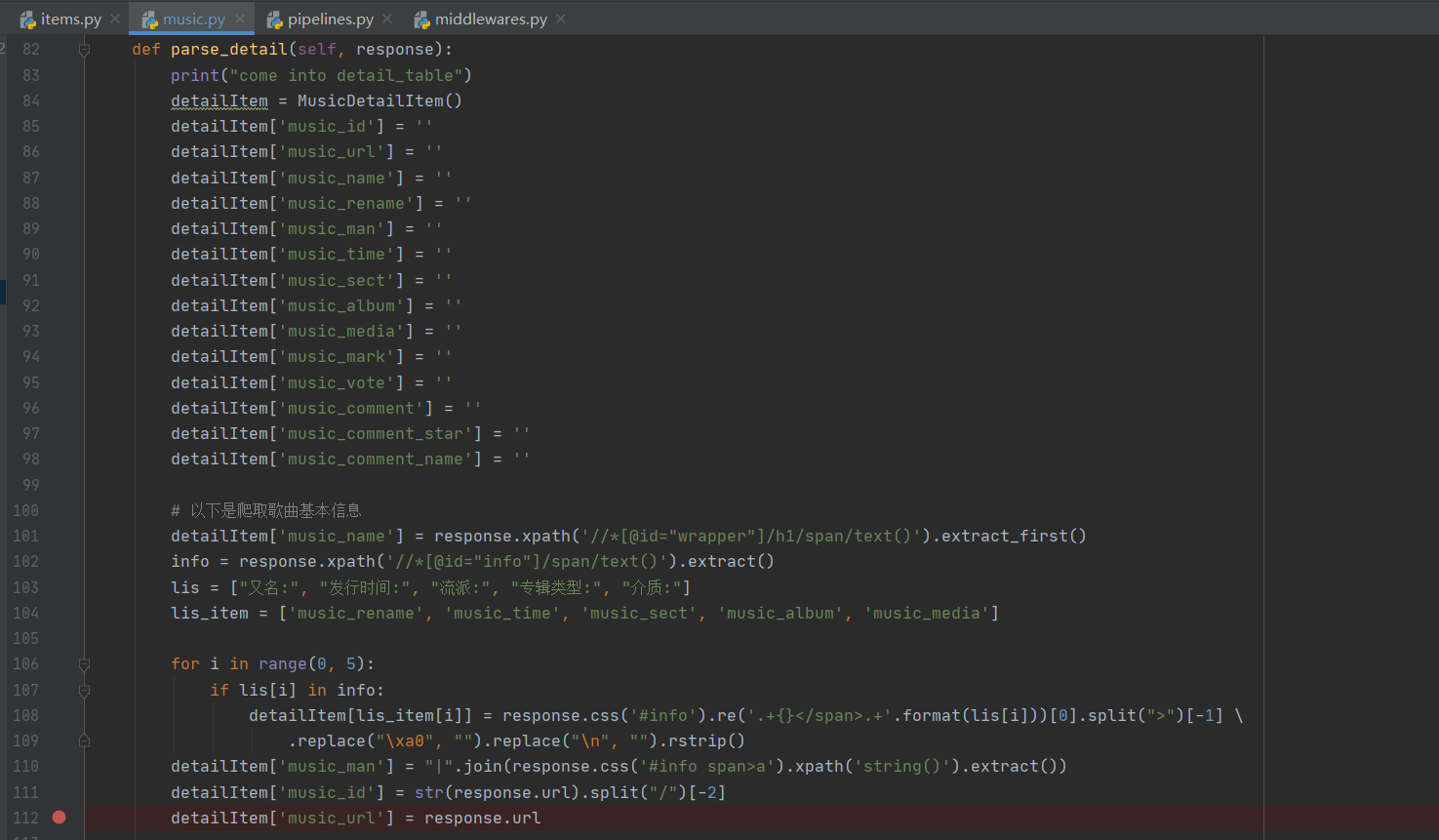
然后正式开始爬取数据，规定从tag页面开始。由于程序从parse函数开始，我们对当前爬取页面进行分类讨论，一次采用不同的处理方式。

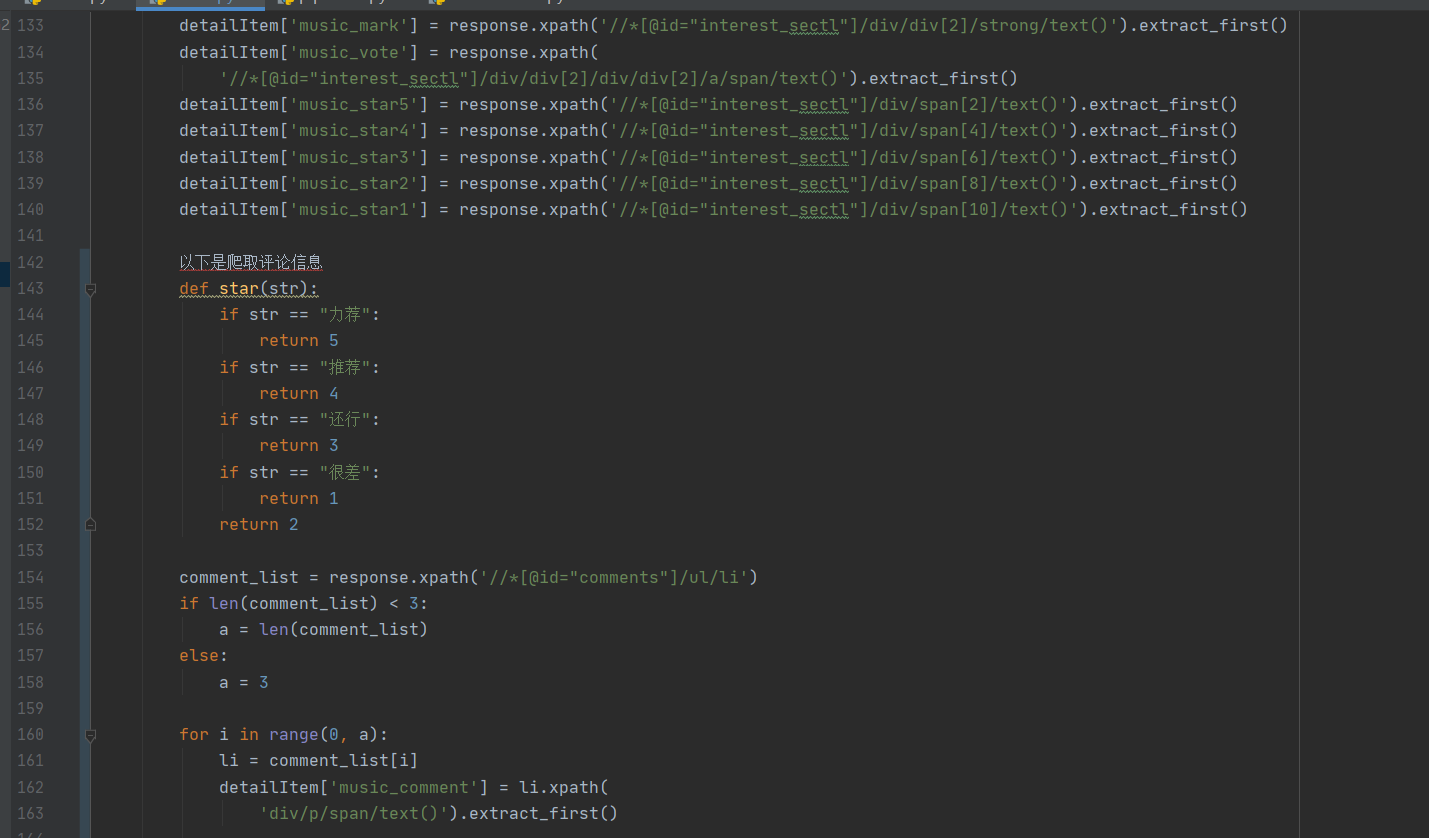


Tag页面我们就便利，将下一级链接压入访问池。



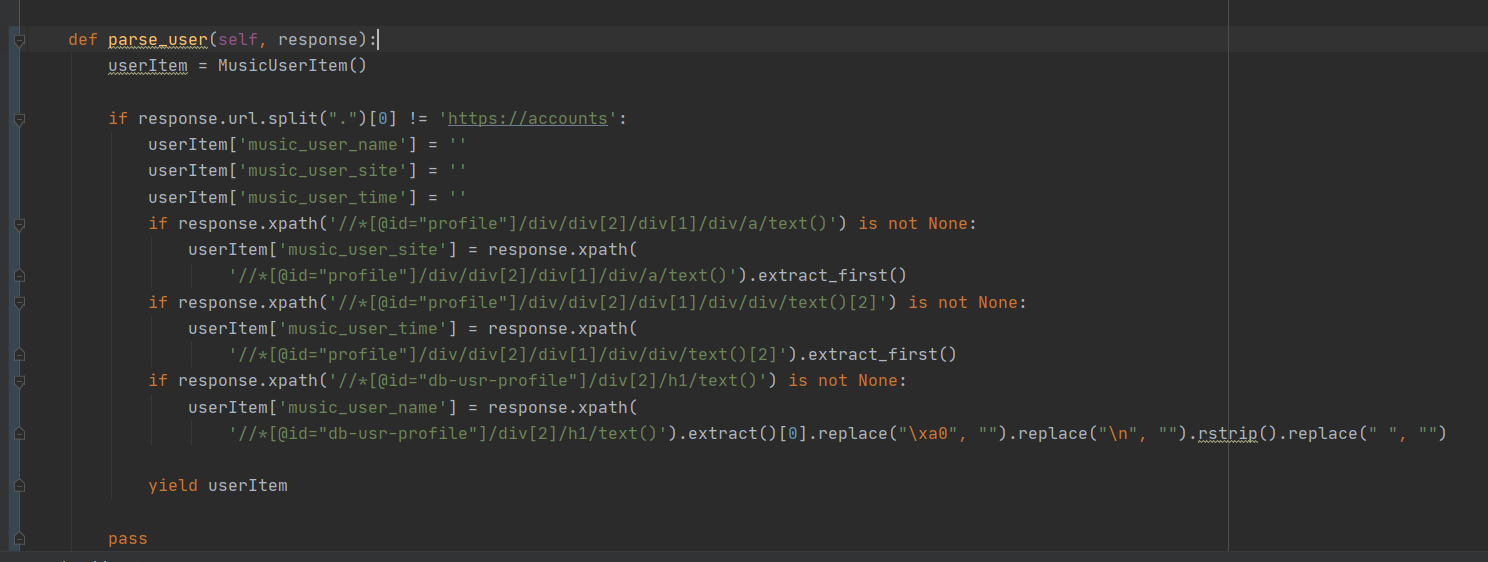
Table页面只读取两条信息，主要是采集音乐详情页链接并压入访问池。

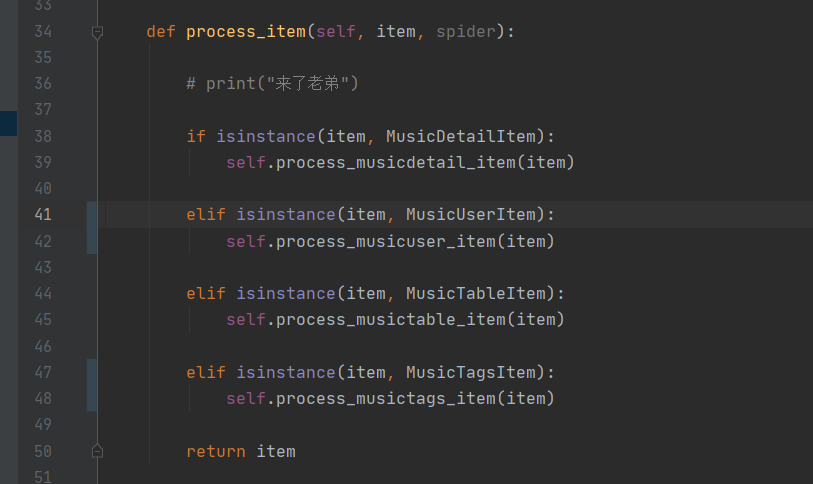






Detail页面要爬取很多数据，只要按部就班，找准路径，提取所需信息即可。

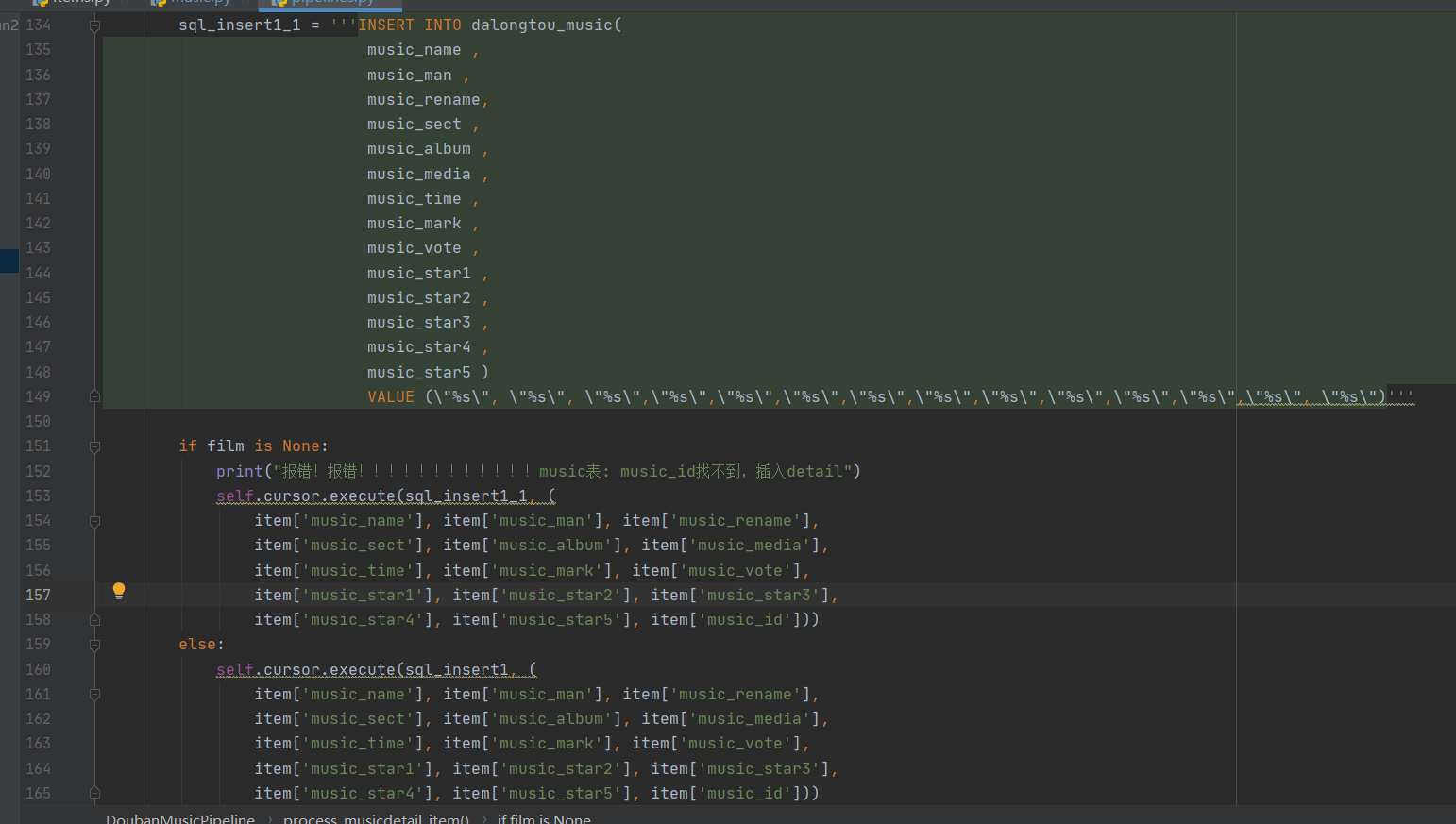
用户界面也很简单，爬取所需信息即可。



由于我们的item是分开的，所以这里我们分开判断即可。

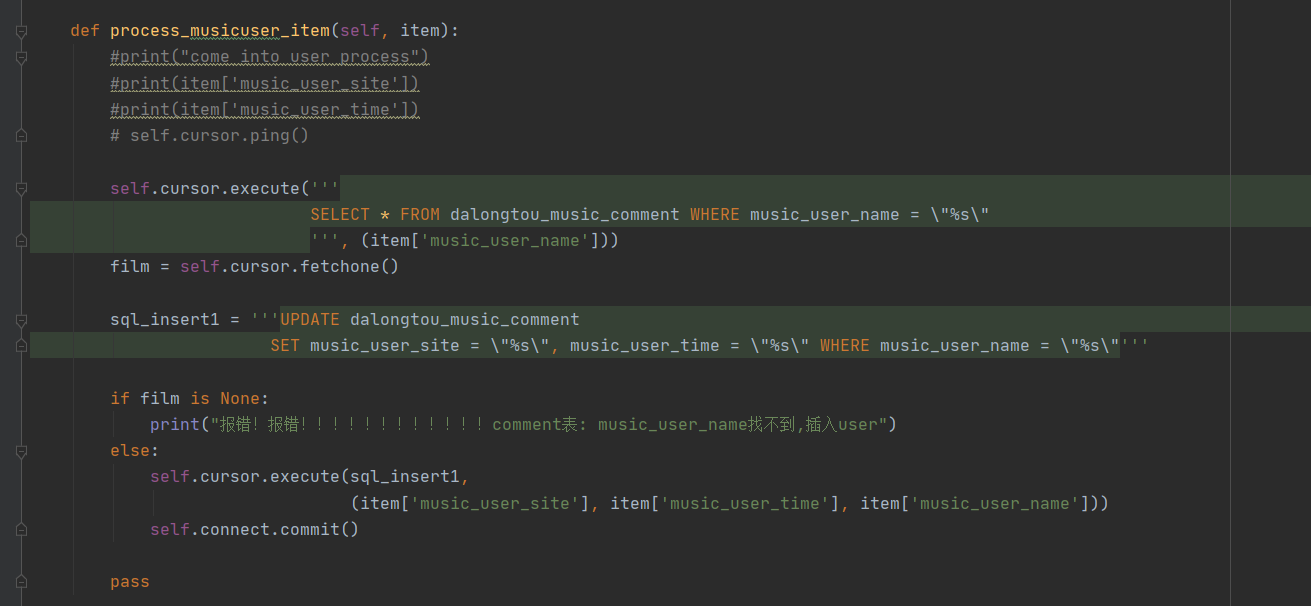


Table页面我们只读取两条信息，然后在数据库中建立新的条目，以便在详情页中对其增添内容。

detail中最复杂，先判断条目是否存在，存在则更新，不存在则插入。



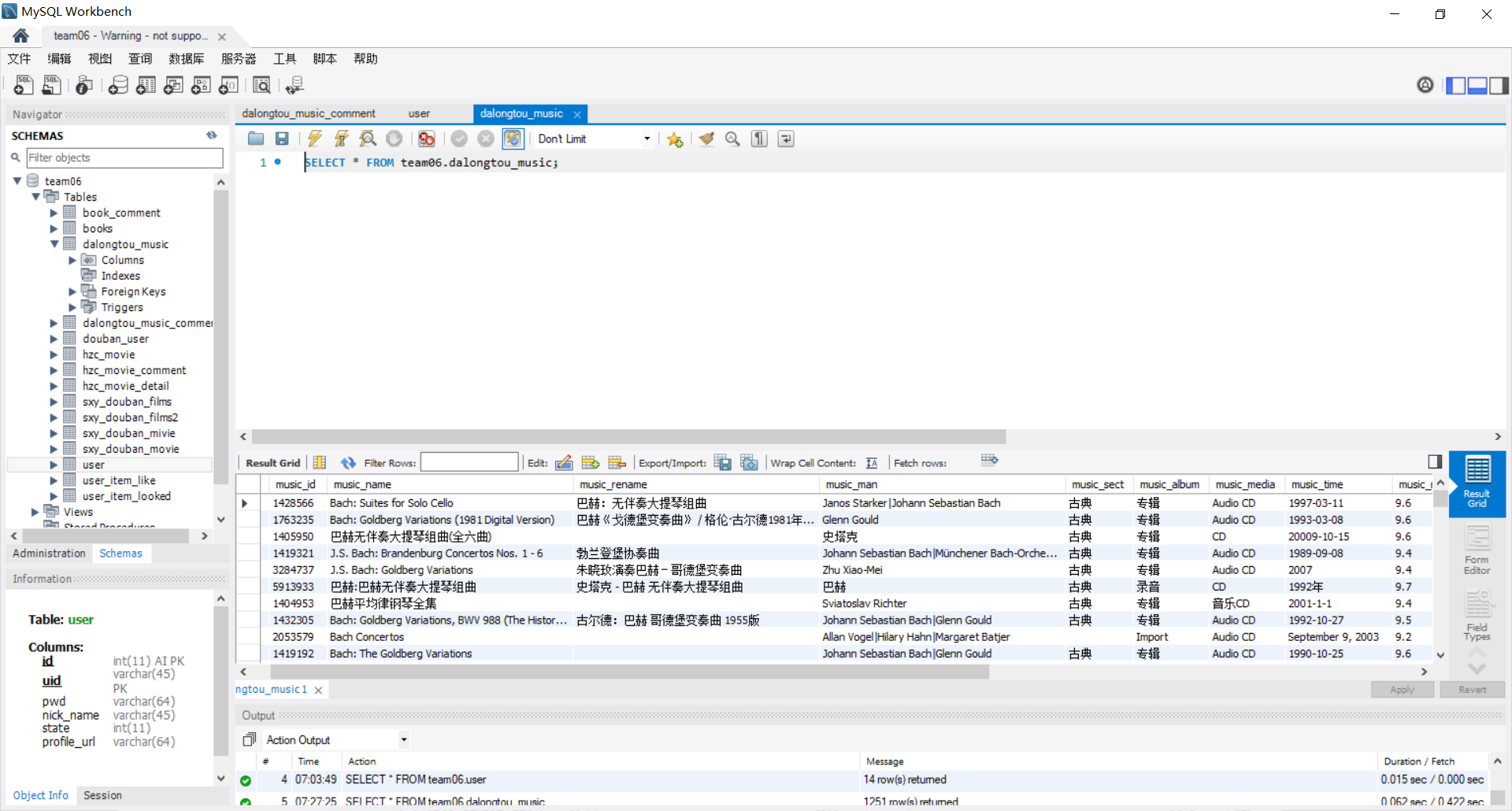
此外还要再comment表中根据音乐id与评论用户名称建立新条目，以便在用户页面读取用户信息对其进行更新。

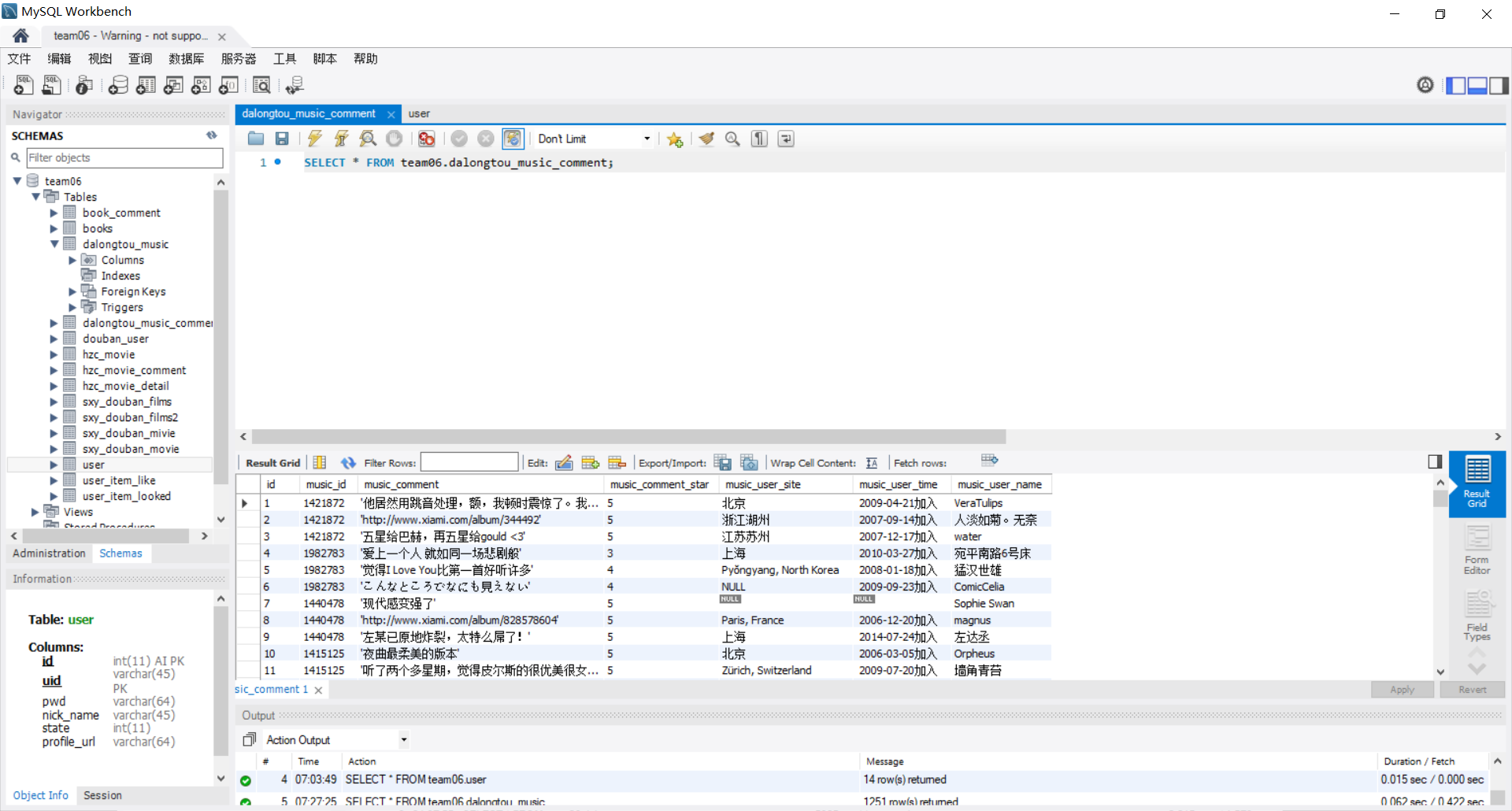


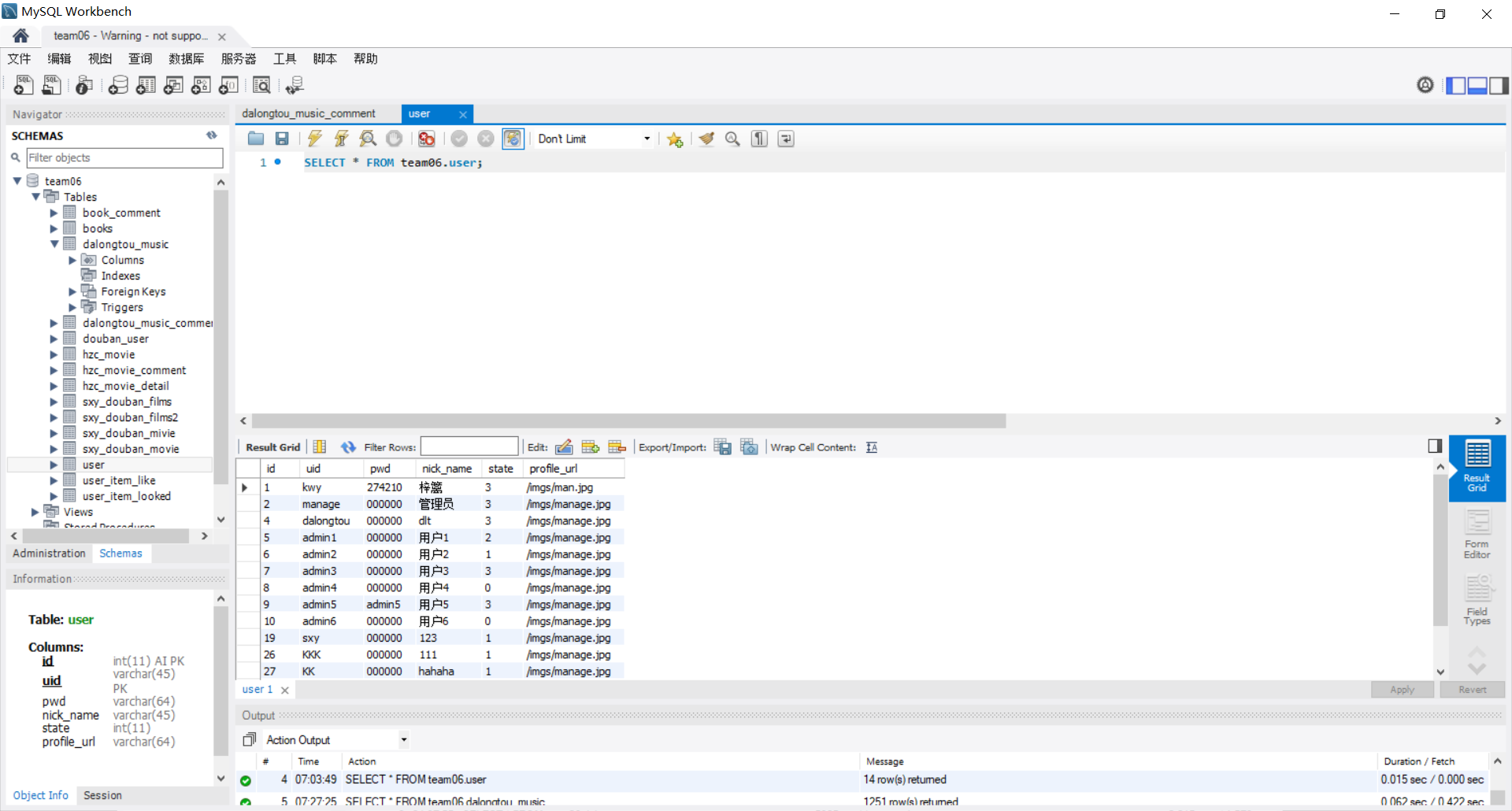
更新用户信息的代码

到这里，信息爬取工作就基本结束，其中还遇到很多报错，但是时至今日，我们都已经完美解决！

以下是数据库表的建立结果：



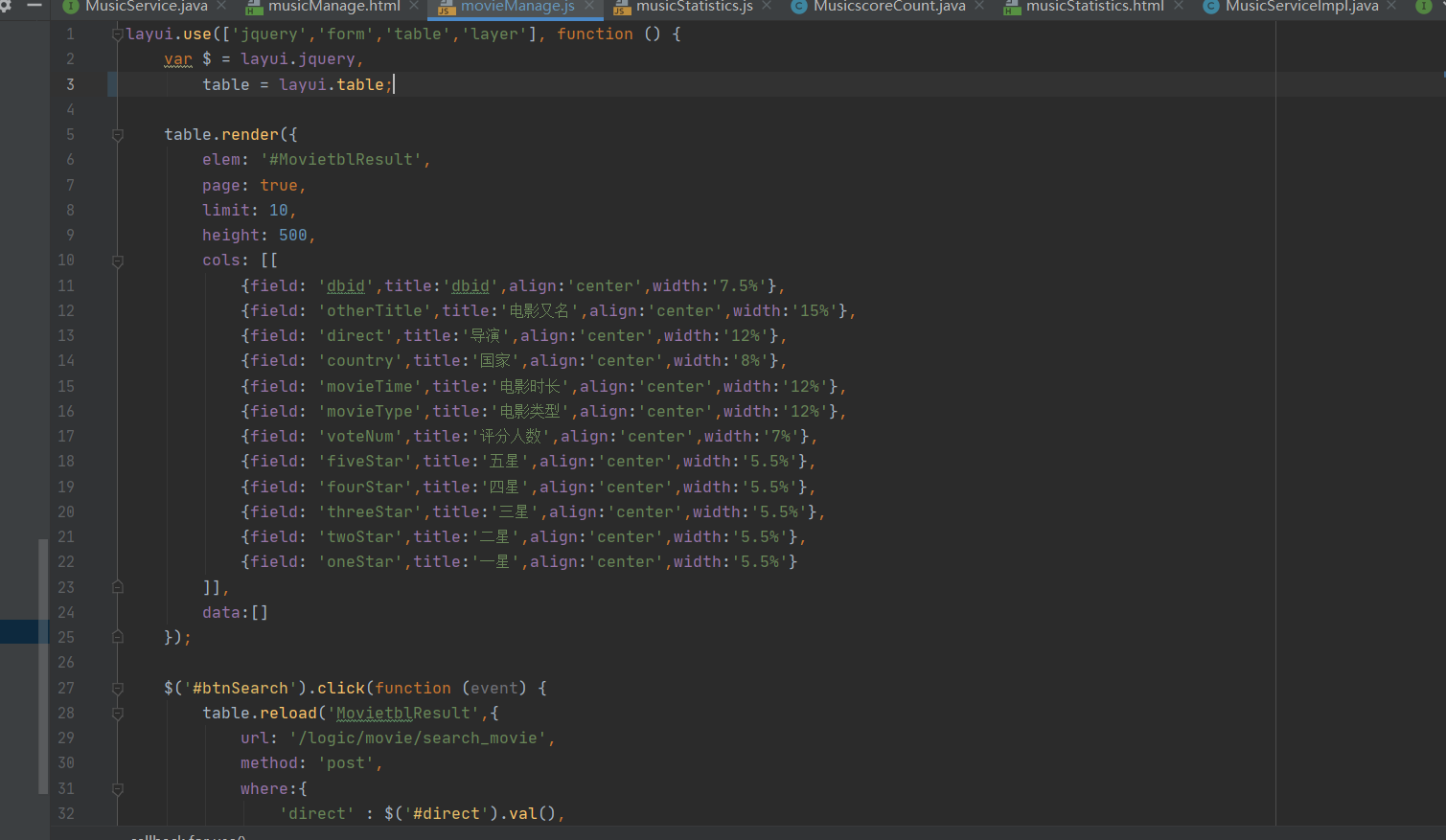




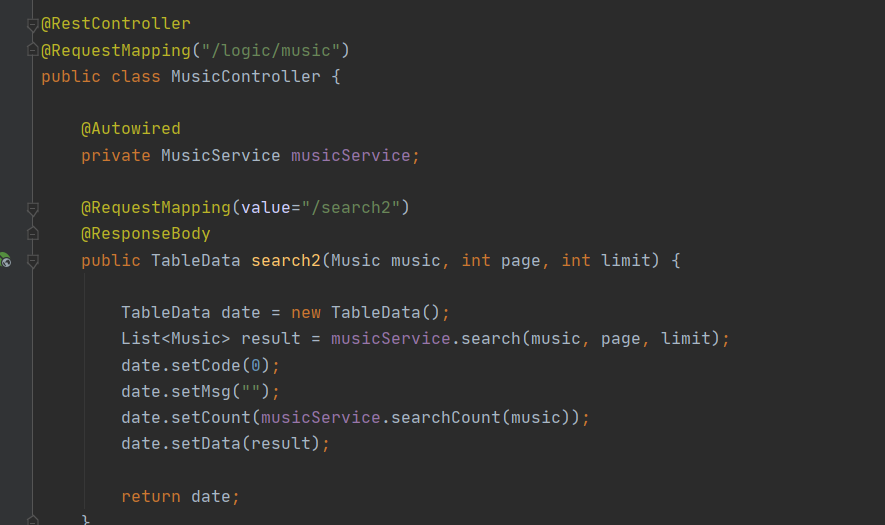
现在我们就进入到java开发阶段！

Java开发大致流程如下：前端html页面->js层->controller层->service层(此处需要实现)->mapper层，mapper层是底层实现层，然后将所得数据一步步返回到前端页面或js层处理后传给前端页面。

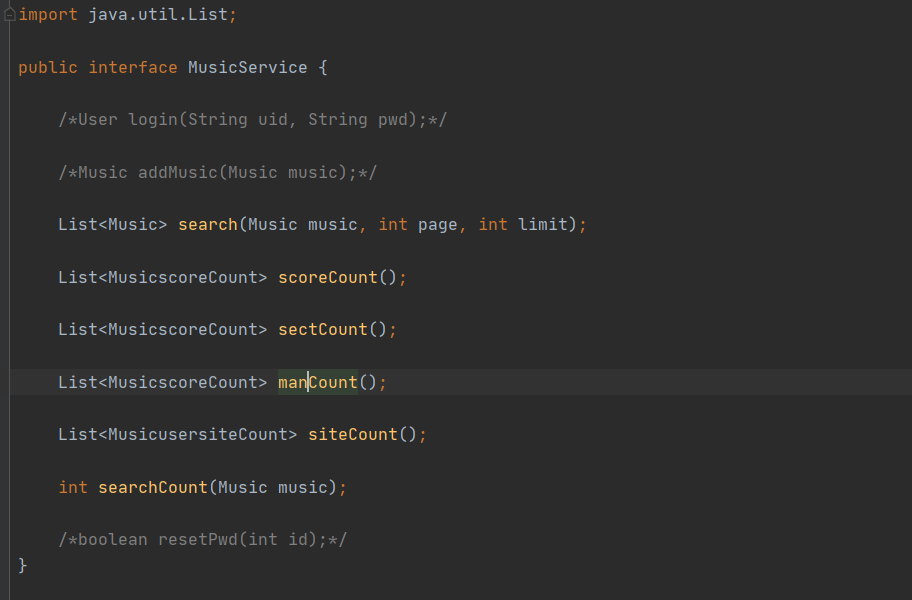
下面我将展示音乐模块的主要代码。



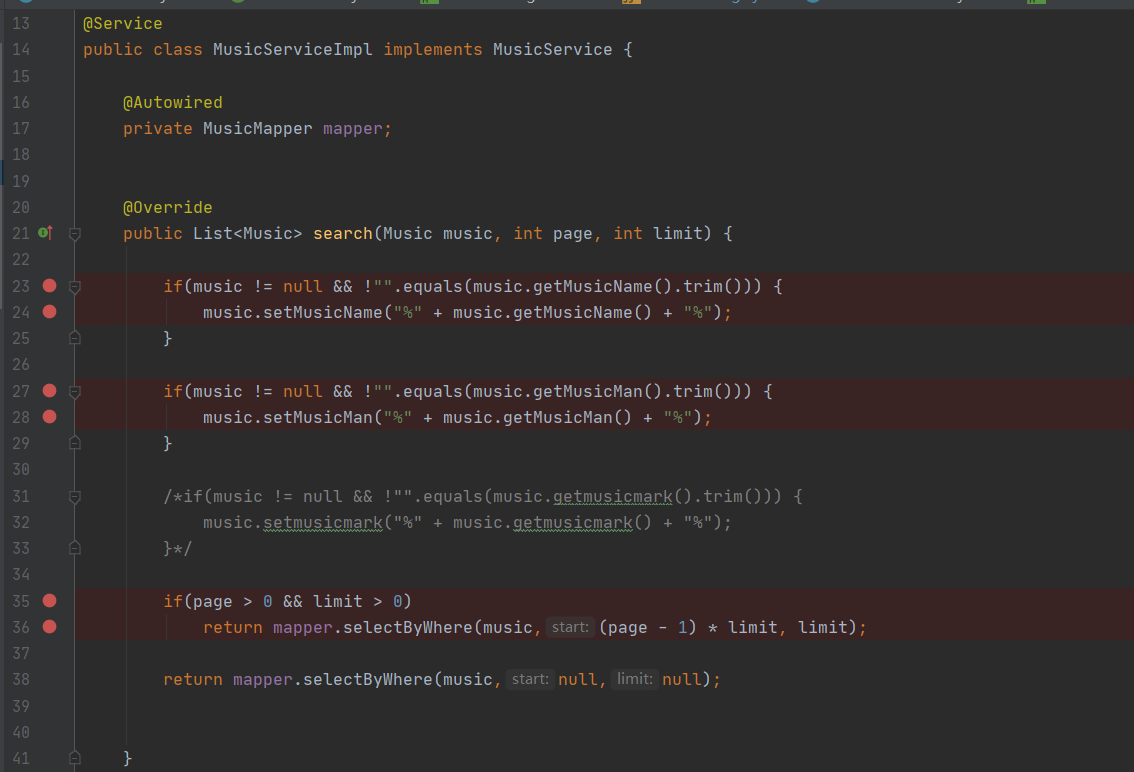
音乐检索js代码。这里我们通过id与前端标签对应，单机按钮后，将前端输入框中数据带入后端进行检索，cols中预留的位置是为了存储从后端传回的数据并显示在页面上。



在controller层中写好方法，通过mapping地址来精准调用。这里我们前往执行service层，这是业务层，在技术方面是无关紧要的，但是作为软件开发的一部在业务层面是十分重要的。



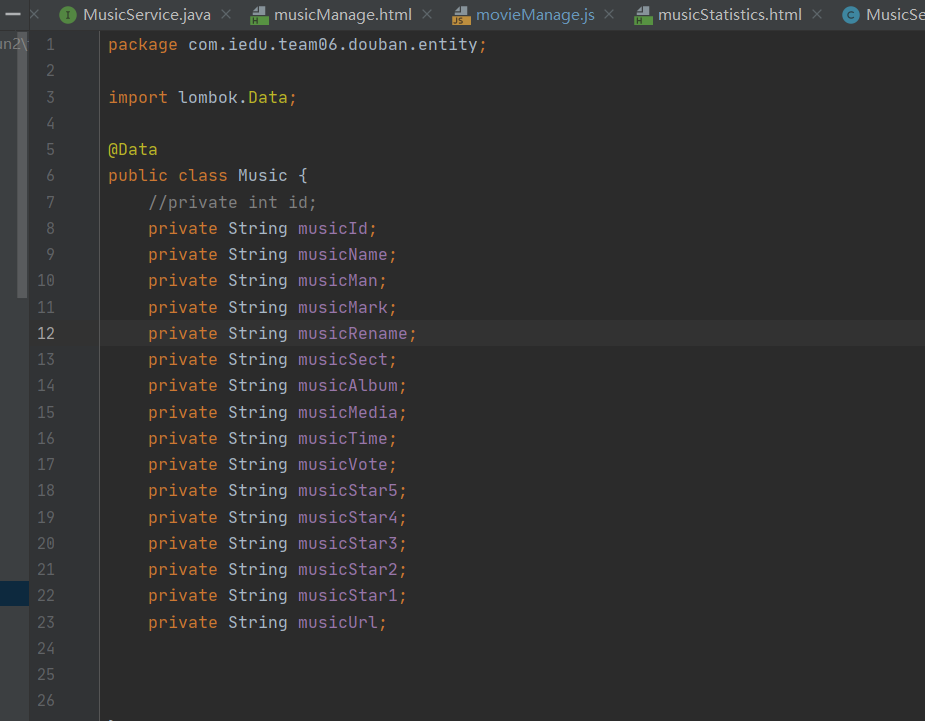
接口很简单。

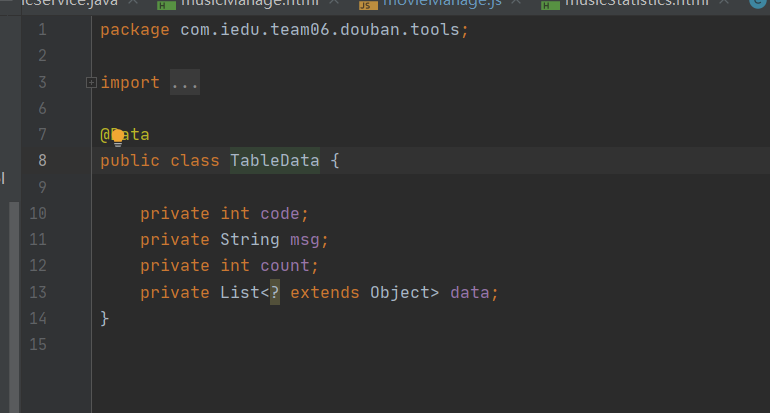


这里就是实现类了。由于我们的检索方式只有三种，所以只需进行三次判断。根据前端传来的信息，这里我们调用mapper层，进行最终的查询。



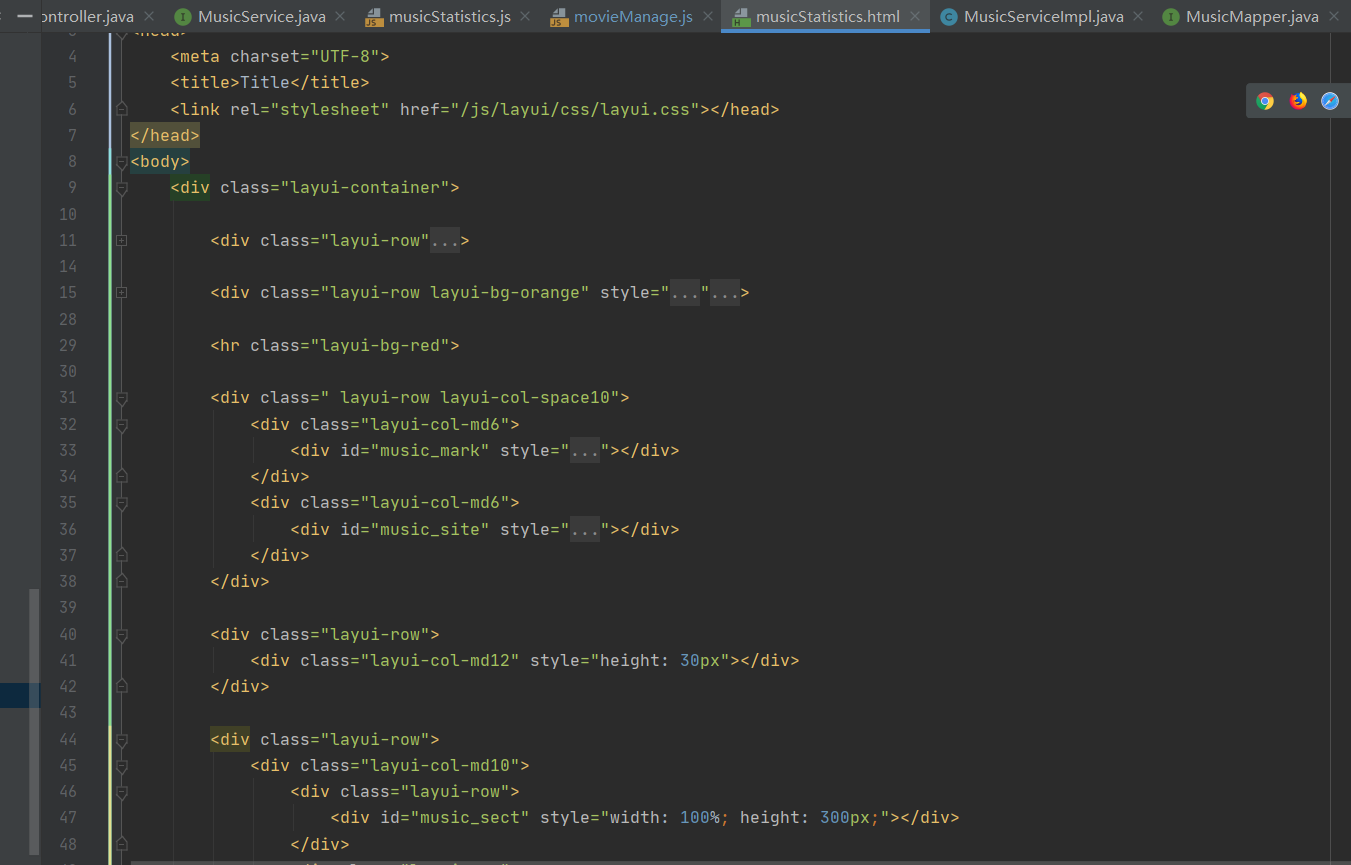
作为实现的最底层，每一行代码都是有着实实在在的功效，通过从前端一路带来的信息，我们在数据库中检索，将检索到的信息再一路传回到js层，这就完成了一次“循环”，我们刚才所见的这只是一条数据通路，除此之外，定义类，工具类等也发挥了巨大的作用。



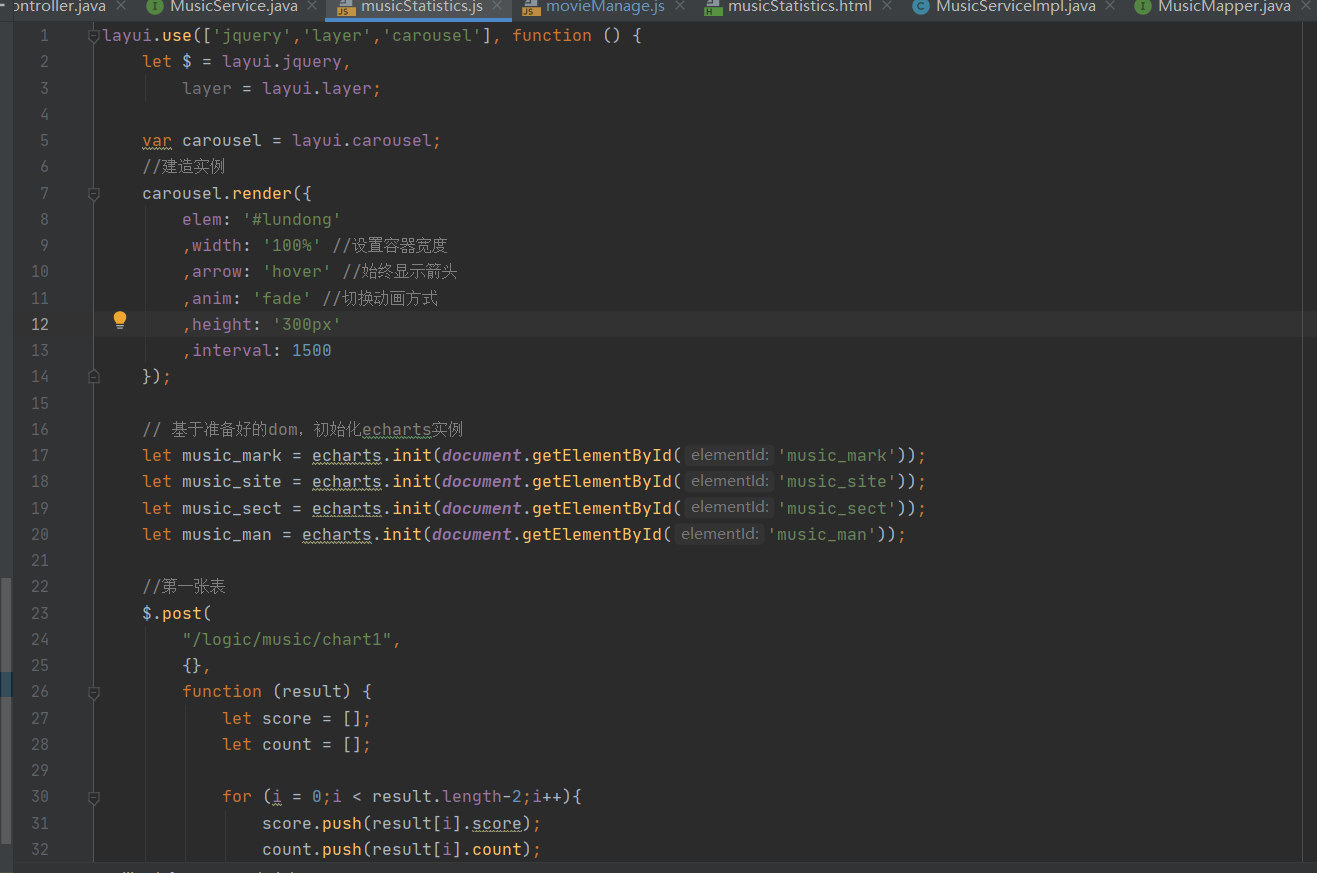


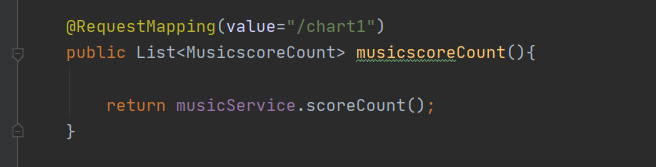
这就是检索功能阐述，此外还有数据分析功能，其中包含多个图表，但都是大同小异，我们以一个为例。

数据分析我们采用前端的图表来进行最直观的表示。我们所用的echarts插件，(支持国产)，辅以layui美化界面。

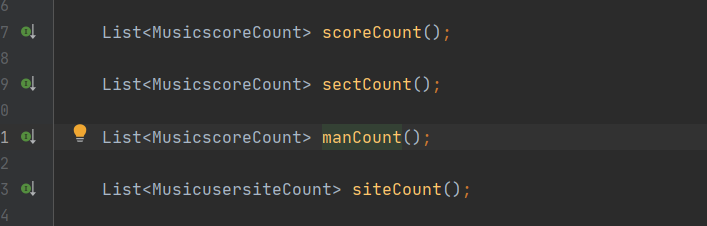


首先为图表规划好空间，我们使用layui的布局方法(12等分法)。

然后再js层为js与前端的数据传输建立条件，依旧使用layui框架。在js层定义图表样式，并为其传值。值从何而来？当然是通过mapping精准定位了。



这里再controller层中调用service层。

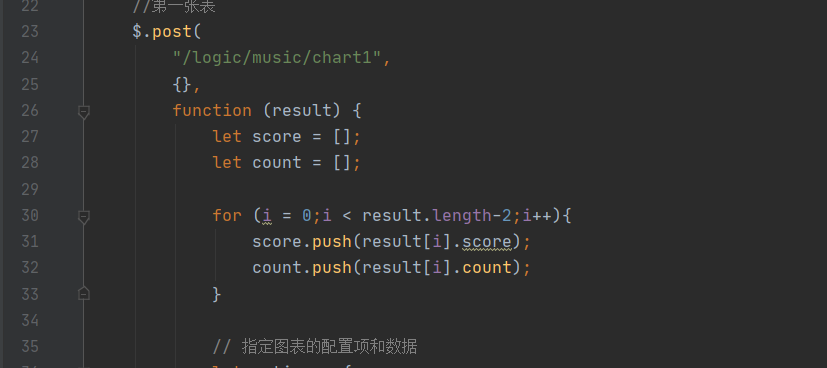




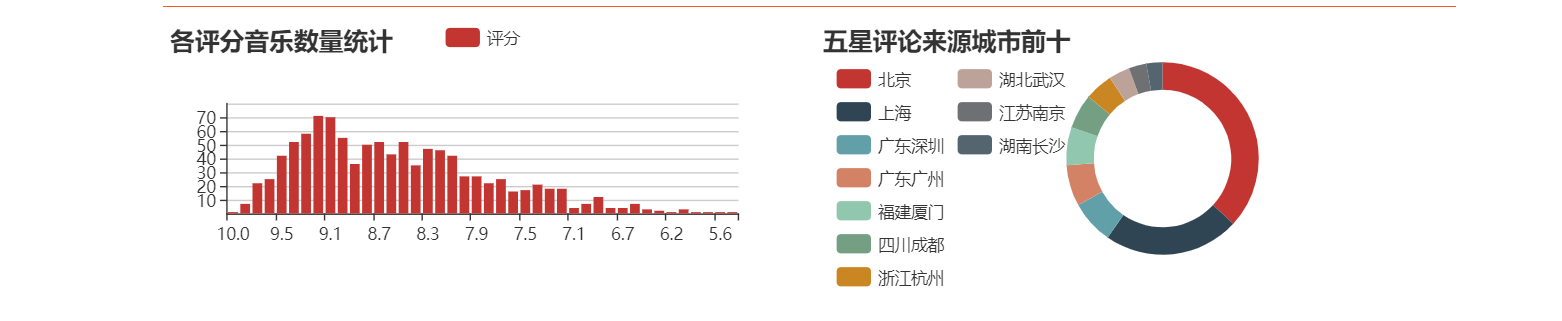
通过service接口找到实现类，在实现类中调用mapper层。



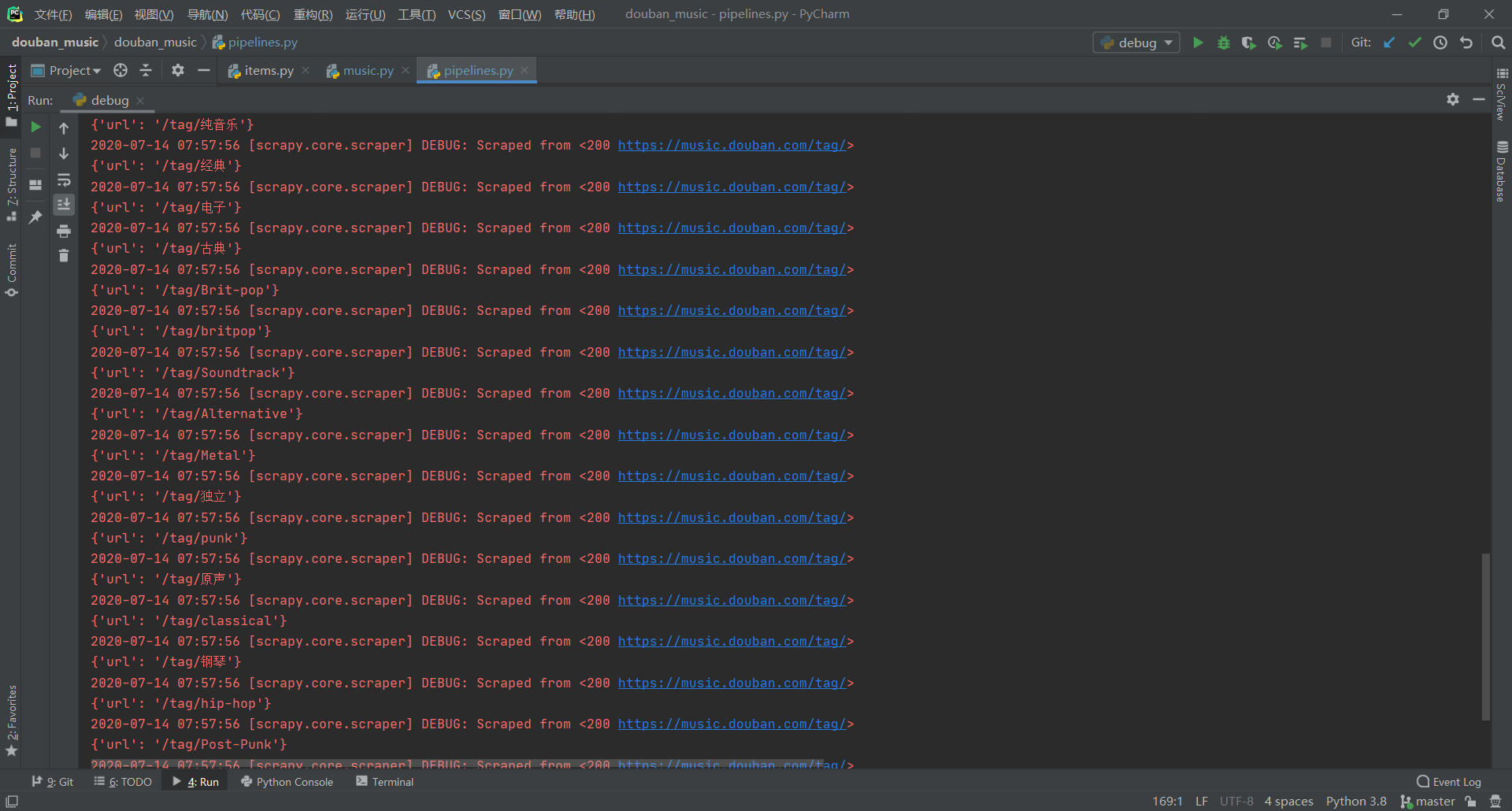
在mapper层中具体实现我们的要求。得到数据后传回。



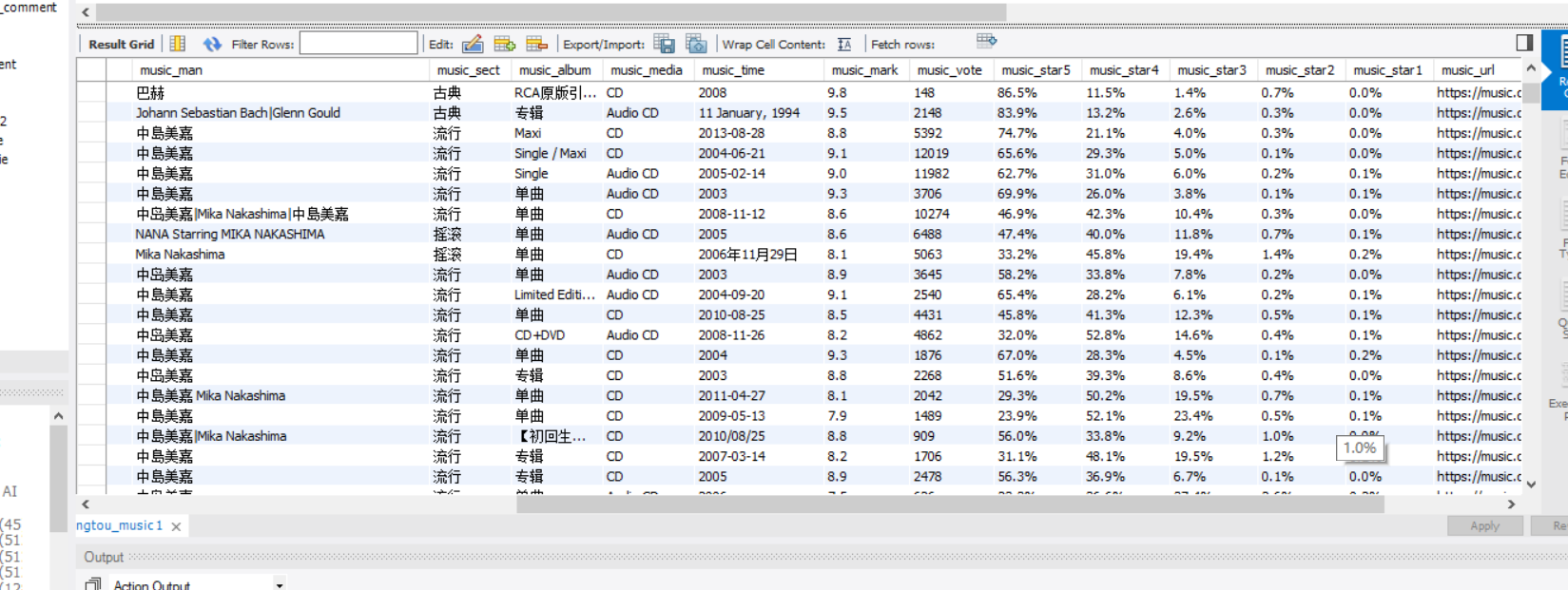
一直到js，我们对数据进行分装并呈递给图表，最后，在前端图标就可以正确的显示了。



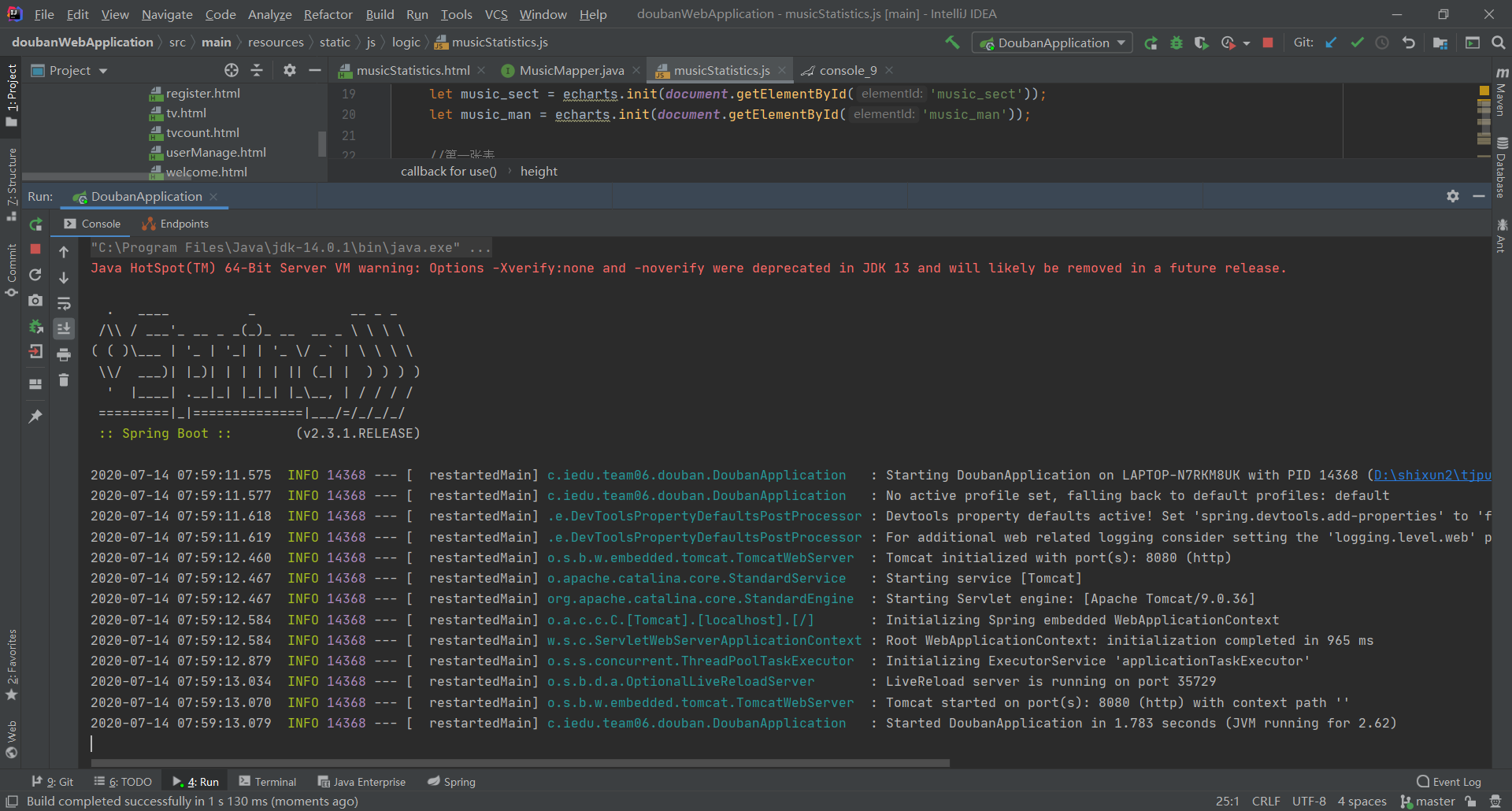
最后，我们再走一遍所有操作流程。



爬取数据。



存入数据库。

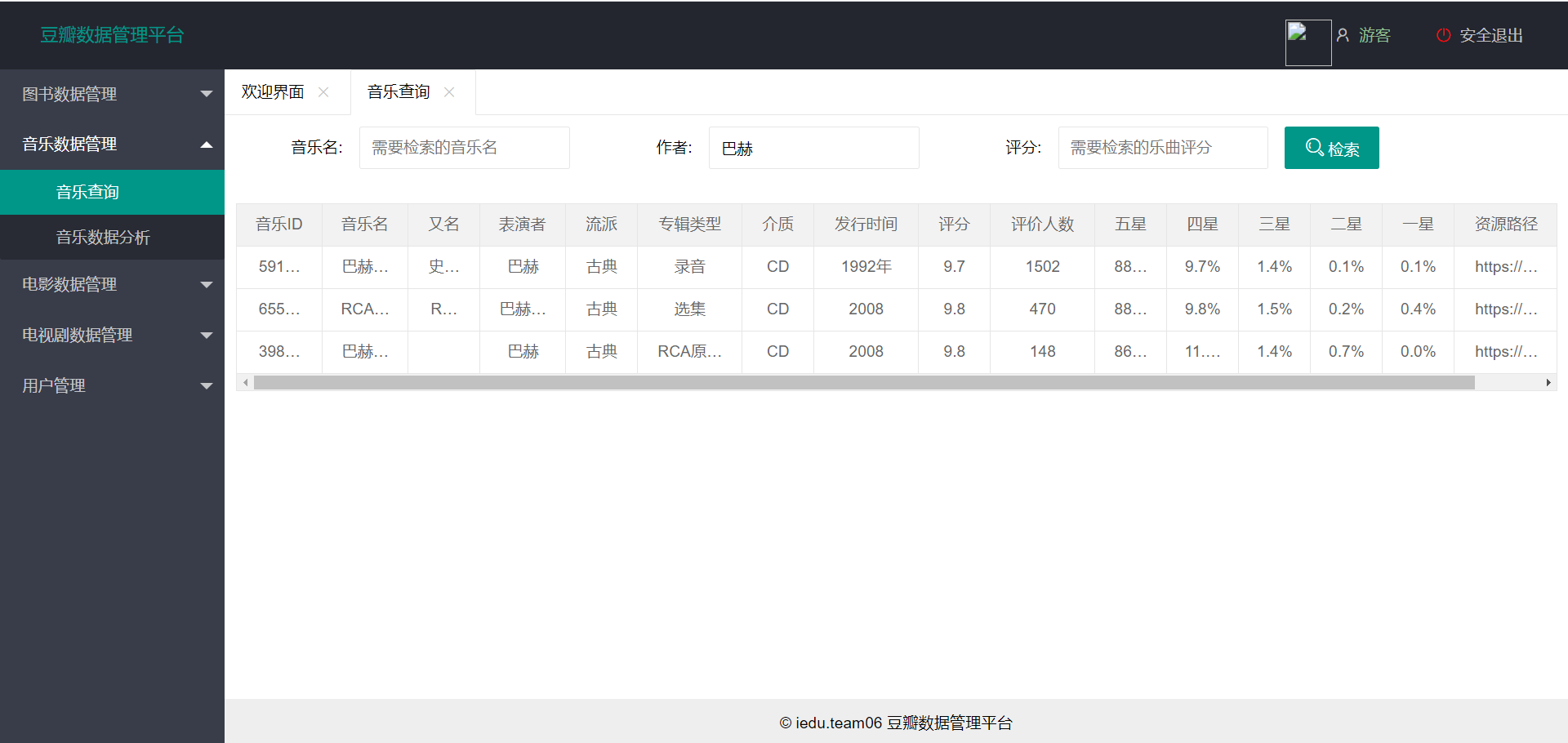


后台服务器启动。

不登录的情况下进入页面。

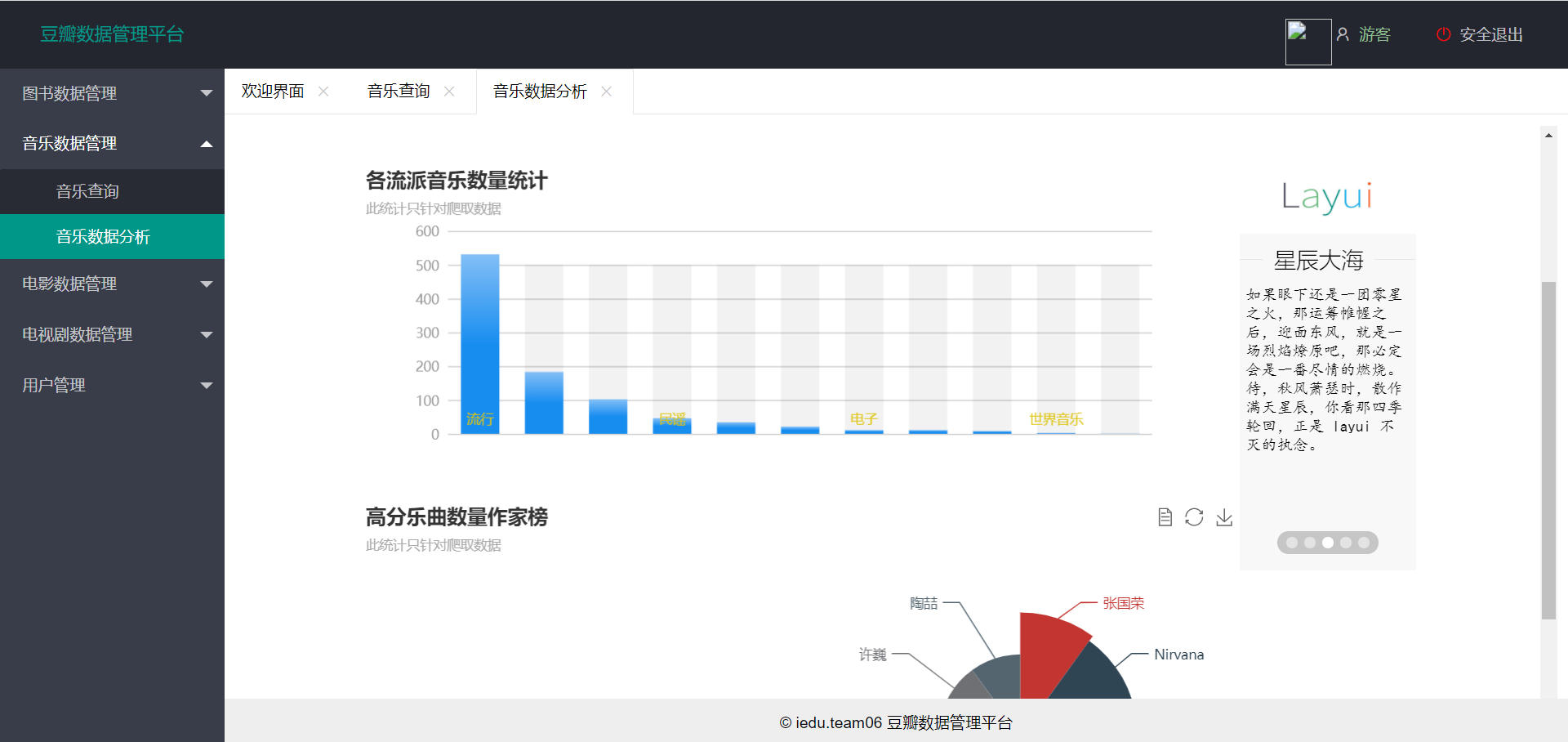


进入查询页面。



查询所需信息。



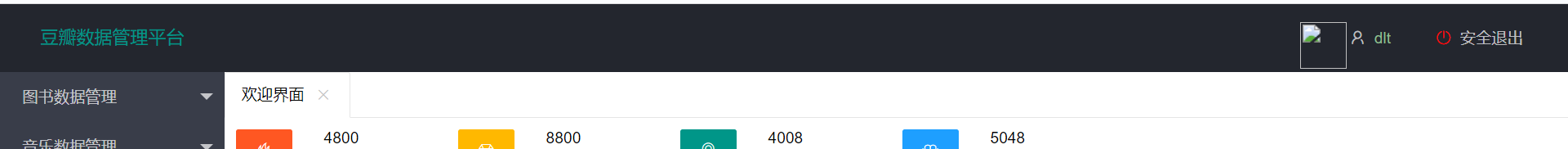


观赏数据分析页面。

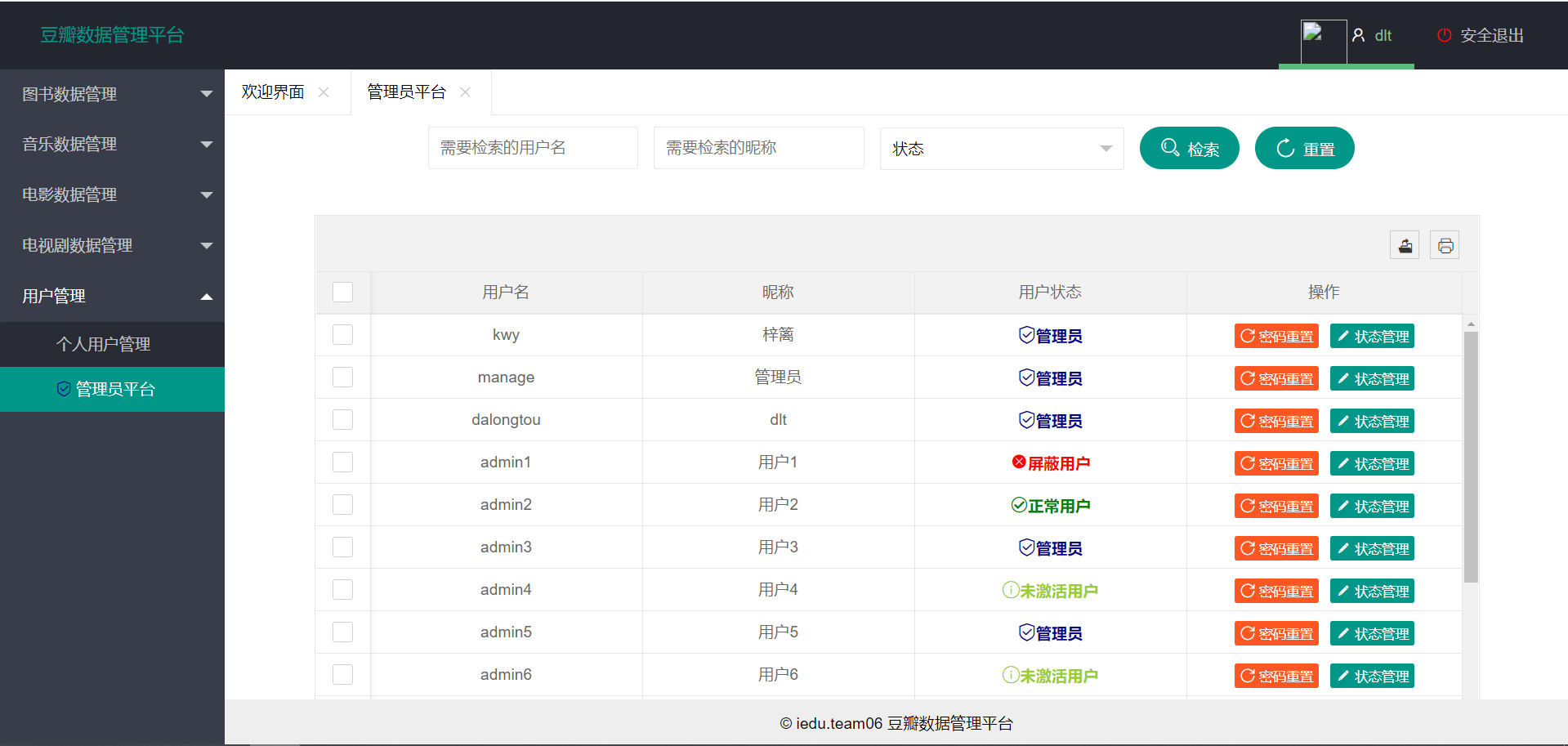


点击‘安全退出’后进入登录页面





输入账号即可登录



作为管理员登陆后还可以检索当前所有注册用户。

**5 系统测试**（小三黑体，居中）

详细写明测试方法、测试描述（bug与修正方法等）、预期结果等；