

毕 业 论 文

|  |  |
| --- | --- |
| **题 目：** | 基于SpringBoot的爱乐在线音乐共享系统的设计与实现 |

|  |  |
| --- | --- |
| **系 部：** | [数据科学与人工智能系](http://cs.imnu.edu.cn/info/1009/1394.htm) |
| **专 业：** | 计算机科学与技术（网络编程） |
| **学 生：** | 杨硕 |
| **学 号：** | 20161104615 |
| **指导教师：** | 张树钧 |
| **日 期：** | 2020年2月20日 |

目 录

[一、绪论 1](#_Toc28186)

[1.1 SpringBoot简介 1](#_Toc13522)

[1.2 项目简介 1](#_Toc26129)

[1.3 可行性分析 1](#_Toc8398)

[1.4 开发环境搭建 2](#_Toc10526)

[二、系统需求分析 2](#_Toc15047)

[2.1 功能需求分析 2](#_Toc18797)

[2.2 性能需求分析 2](#_Toc18503)

[2.3 系统流程分析 3](#_Toc10825)

[三、系统总体设计 3](#_Toc3012)

[3.1系统功能模块分析 3](#_Toc10045)

[3.2系统类及其关系 3](#_Toc20939)

[3.3用例图 4](#_Toc25891)

[四、开发环境 4](#_Toc5188)

[4.1硬件平台 4](#_Toc9434)

[4.2软件平台 4](#_Toc17203)

[4.3环境配置 5](#_Toc13043)

[五、系统详细设计 5](#_Toc13418)

[5.1数据库设计 5](#_Toc31307)

[5.2 Bean实体类设计 6](#_Toc27966)

[5.3 Mapper接口 6](#_Toc6428)

[5.4 Service接口 9](#_Toc23276)

[5.5 Controller方法 12](#_Toc15543)

[六、测试与调试 20](#_Toc24323)

[七、总结 21](#_Toc32168)

[参考文献 22](#_Toc13801)

基于SpringBoot的爱乐在线音乐共享系统的设计与实现

计算机与信息工程学院 2016级网络编程班 杨硕 20161104615

指导教师 张树钧

摘要 音乐总是伴随在生活的左右，而科技和音乐的融合便是不断改进和完善当前音乐网站对旧时音乐播放、共享功能的实现，这在一定程度上实现了人民对于美好生活的需要，同时也促进了音乐行业的发展。在线音乐播放系统的实现使得音乐播放功能不拘束于特定得播放设备，音乐的存储介质不拘泥于CD光盘、磁带、录像带等实体形式，便于随时随地的进行音乐播放和分享。在这里我将采用SpringBoot框架、Mybaits和BootStrap框架设计实现该系统，开发环境使用IDEA。界面操作容易，以带给用户极简的使用体验。

关键词 音乐；SpringBoot框架；Mybaits；BootStrap框架；

1. 绪论

1.1 SpringBoot简介

SpringBoot是一个轻量级的开源框架，它基于Spring4.0设计实现它在继承Spring框架的优秀特性之余，还简化了配置，解决了依赖包之间的冲突，使得整个应用在搭建、开发时更加便捷、稳定。所以本项目中就以SpringBoot2.2.1框架进行开发。以便于更好的兼容其他依赖包。

1.2 项目简介

近几年，各类音乐类、视频类网站迅速崛起，并走红市场。在促进音乐、综艺等行业发展的同时，也丰富了人们的文化生活，增加了趣味。可见这类型的网站贴近群众，接地气，更容易被人们理解和接受。作为一名计算机学生，在毕业之际，我选择去做音乐播放网站，在日常的生活中也可以使用。项目取名为IMusic，意味爱音乐、爱生活，希望音乐可以伴随生活，丰富生活。

1.3 可行性分析

本项目是由IDEA集成SpringBoot为开发平台进行在线音乐网站的开发。IDEA是以java编程语言开发的集成环境，在业界被公认为最好的java开发工具，尤其在智能代码助手、代码自动提示、重构、J2EE支持、各类版本工具等方面的功能可以说是超常的。同时SpringBoot整合了Thymeleaf模板引擎，在前端页面的显示、排版上具有很好的支持。Thymeleaf的语法更加接近于HTML，并且有不错的扩展性，更适合前端的开发使用。使用SpringBoot+Mybaits的组合可以更加灵活的操作数据库，方便项目的开发。前端页面则采用简洁、直观、强大的BootStrap框架，可以很好的集成JQuery等优秀插件。作为音乐播放网站最重要的播放插件——JPlayer插件，它支持多种平台和浏览器，支持多种媒体播放，具有很强的兼容性。

1.4 开发环境搭建

开发环境的搭建分为本地服务器的搭建和项目的搭建。由于JPlayer插件不支持直接访问本地路径，所以需要采用搭建本地服务器的方式访问本地路径，在CSDN博客中搜索“JPlayer访问本地音乐路径”可以找到。本地服务器采用Tomcat搭建本地服务器。

1. 系统需求分析

2.1 功能需求分析

游客访问网站主页，显示各音乐信息。

用户登录网站后，显示网站主页信息。任意点击音乐可以播放该音乐。用户在自己的主页中可以上传音乐。

管理员登录网站后，显示后台页面。可对用户的个人信息和积分进行管理。可对网站主页中的音乐信息进行管理。可上传、发布、下架音乐。

2.2 性能需求分析

（1）及时性

本项目是以浏览器为使用平台的，所以对于用户的操作必须做出及时的响应，而且画面更新要流畅。

（2）易操作性

对于在线音乐播放网站来说，需要实现音乐播放的易操作性。易操作性包括操作的简洁、易懂。

1. 存储性

本项目采用本地服务器，所以有具有较高的存储性，满足用户上传、访问等需求。

2.3 系统流程分析

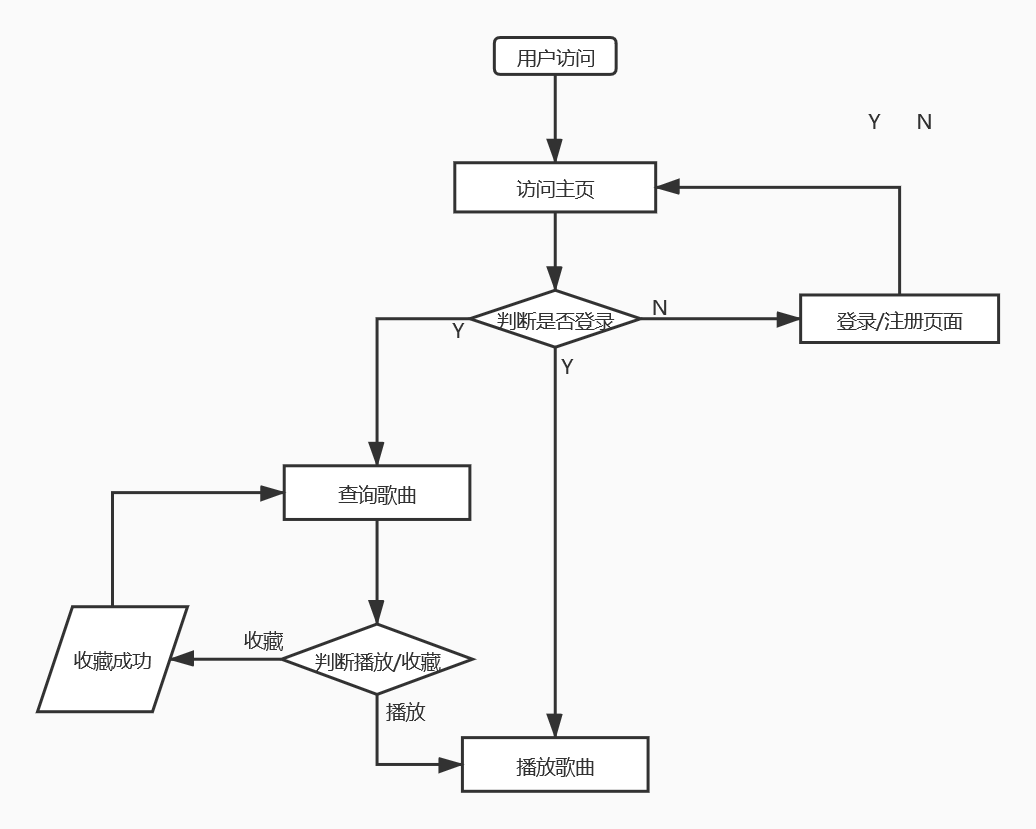
根据系统操作流程，制作出如下流程图：

图2-3-1流程图

1. 系统总体设计

3.1系统功能模块分析

该系统主要是以SpringBoot框架为基础，采用MVC的设计理念，从而实现在线音乐播放的各项功能。以Controller为划分，可分为五大不同模块：

1. UserController用来完成用户基本操作，如登录、注册、修改、收藏、分享、播放功能。
2. MusicController用来完成歌曲查找、用户收藏列表查找、歌曲列表数据组装功能。
3. FilesController用来完成音乐、图片的文件上传，修改音乐、图片的上传文件以及文件的下载功能。
4. AdministratorController用来完成管理员对用户的查询、更新、删除和添加功能，以及对音乐的查询、上传、更新和删除功能。
5. LoginController用来完成游客访问的权限控制和用户退出功能。

3.2系统类及其关系

系统的config类、mapper类、service类和controller类的调用关系如图3-2-1所示：

**前端请求**

**Config判断状态**

**Servlet接收**

**调用Service**

**调用Mapper**

Form表单

Config

Servlet

Service方法

图3-2-1

Mapper

3.3用例图

分享

修改

查找

删除

增加

用户信息

音乐信息

People

Administrator

User

Tourists

身份验证

图3-3-1

1. 开发环境

4.1硬件平台

1. CUP：i5-8265U
2. 内存：8.00GB

4.2软件平台

1. 操作系统：Windows10
2. IDE：IDEA、Navicat、Tomcat8、HBuilder X
3. 数据库：MySQL

4.3环境配置

1. Java：Java8
2. MyBatis：2.2.1
3. Druid：1.1.18
4. 系统详细设计

5.1数据库设计

数据库选用MySQL，可以更好的和Java进行交互，同时实现对数据库中表的高效操作。

1. User表设计

User表中设置九个字段，令user\_id字段为主键，并设置自增，用于标识用户，同时方便管理员对用户进行操作，提高针对性、顺序性和可读性。同时设置user\_Administrator字段，用于区分管理员和普通用户，从而实现不同身份不同操作。设置user\_phone字段来验证用户的请求等相关操作。设user\_birthday字段，实现用户的多种方式登录。

1. Music表设计

Music表中设置六个字段，令music\_id字段为主键，并设置自增，用于标识音乐，同时方便用户、管理员对音乐的操作。设置music\_storagepath字段来存放歌曲的路径。设置music\_img来存放歌曲封面的路径。因为歌曲或歌手存在名称较长或人员较多的情况，所以music\_name和music\_singer两字段长度均为500，最大可能的包含上述情况。提现较强的存储性和容纳性。

1. Favoritesongs表设计

Favoritesongs表设置四个字段，令id为主键，并设置为自增，便于标识。同时设置music\_id和user\_phone为外键，将user表和music表联系起来，实现用户收藏歌曲的记录功能。

1. Sharesongs表设计

Sharesongs表设置五个字段，令id为主键，并设置为自增，便于标识。同时设置music\_id和user\_phone为外键，将user表和music表联系起来，实现用户分享歌曲的记录功能。其他字段便于显示相关信息时调取。

1. Comments表设计

Comments表设置个字段，令id为主键，并设置为自增，便于标识。同时设置music\_id和user\_phone为外键，将user表和music表联系起来，实现用户评论的记录功能。其他字段便于显示相关信息时调取。

5.2 Bean实体类设计

实体类是一个公共类，包含UserBean、MusicBean和[FavoriteSongs。](https://github.com/20161104615/GraduationProject/blob/master/music-day01/src/main/java/com/ys/demo/bean/FavoriteSongs.java" \o "FavoriteSongs.java)对应于数据库中user表、music表和Favoritesongs表。

1. UserBean的设计

设置对应于数据库user表中各字段的变量，并设置为私有变量，实现变量的封装，同时为每一个变量设置Get()和Set()方法，用于实现对变量的赋值和修改操作。为便于打印输出各变量赋值的情况，重写toString()方法，实现上述功能。为了查看各变量的类型，重写equals()和hashCode()方法，用于实现上述功能。为了UserBean更方便的创建，对构造方法进行重载。

1. MusicBean的设计

设置对应于数据库music表中各字段的变量，并设置为私有变量，实现变量的封装，同时为每一个变量设置Get()和Set()方法，用于实现对变量的赋值和修改操作。为便于打印输出各变量赋值的情况，重写toString()方法，实现上述功能。为了查看各变量的类型，重写equals()和hashCode()方法，用于实现上述功能。为了MusicBean更方便的创建，对构造方法进行重载。

1. FavoriteSongs的设计

设置对应于数据库FavoriteSongs表中各字段的变量，并设置为私有变量，实现变量的封装，同时为每一个变量设置Get()和Set()方法，用于实现对变量的赋值和修改操作。为便于打印输出各变量赋值的情况，重写toString()方法，实现上述功能。为了查看各变量的类型，重写equals()和hashCode()方法，用于实现上述功能。为了FavoriteSongs更方便的创建，对构造方法进行重载。

5.3 Mapper接口

Mapper包下包含两个接口，分别为UserMapper和MusicMapper。分别实现对用户和音乐的数据库操作。

1. UserMapper接口

在UserMapper接口中，使用注解开发的方式，实现对数据库user表的操作。其中包含七种查询方法，分别为：

1. getUserByuser\_phone：通过传入的user\_phone查找user表中的用户信息。
2. finduser：通过传入的UserBean对象查找user表中相应用户的信息。
3. finduserbystring：通过传入的user\_name查找user表中对应的用户信息。
4. findAllUser：通过传入的user\_Administrator查找user表中相应用户的信息。
5. FINDUSER：通过传入的user\_id、user\_phone、user\_email，以及默认user\_Administrator为FALSE的条件，查找user表中相应的用户信息。
6. FINDUSER2：通过传入的user\_id、user\_phone、user\_email，和user\_Administrator，查找user表中相应的用户信息。
7. FINDUSERBYONE：通过传入的user\_phone和默认user\_Administrator为FALSE的条件，查找user表中相应的用户信息。

包含两种插入方法，分别为：

1. insertUser：用于通过输入user\_name、user\_phone、user\_pwd、user\_email和user\_Administrator，向数据库user表中插入一条数据。
2. INSERTUSER：用于通过输入user\_name、user\_phone、user\_pwd、user\_email和user\_Administrator，向数据库user表中插入一条数据。

包含两种修改方法，分别为：

1. updateuser：用于通过输入user\_name、user\_pwd、user\_birthday和user\_introduced，修改数据库user表中已存在的内容。
2. AUPDATEUSER：用于通过输入user\_id、user\_name、user\_pwd、user\_birthday和user\_introduced，修改数据库user表中已存在的内容。

包含一种删除方法，DELETEUSER方法，用于管理员删除user表中某个用户的信息。

1. MusicMapper接口

在MusicMapper接口中，使用注解开发的方式，实现对数据库music表的操作。其中包含九种查询方法，分别为：

1. findMusicBeanList：用于从music表中查找所有的音乐信息。并以ArrayList的形式返回。
2. findMusicByName：通过输入的MusicBean，使用模糊查找的方式查找music表中music\_name相对应的音乐信息。
3. findMusicBySinger：通过输入的MusicBean，使用模糊查找的方式查找music表中music\_singer相对应的音乐信息。
4. accurateFindMusic：通过输入的music\_name，获取查询结果的第一条数据。
5. findMusicByUserPhoneOfFavorite：通过输入的user\_phone，先在favorite表中获取对应用户所添加的音乐id，再通过music\_id在music表中查询该音乐的所有信息。
6. FINDMUSICBYNAME：通过输入的music\_name，使用模糊查找的方式查找music表中music\_name相对应的音乐信息。
7. FINDMUSICBYID：通过输入的music\_id，以歌曲id查找music表中相对应的音乐信息。
8. SHARE\_SONGS\_ARRAY\_LIST：通过输入的user\_phone，来查询sharesongs表中用户的歌曲分享信息。
9. CONMMENTS\_ARRAY\_LIST：通过输入的music\_id，查询comments表中相应歌曲的评论信息。

包含四种插入方法，分别为：

1. uploadMusic：通过输入music\_name、music\_singer、music\_img、music\_storagepath，向数据库music表中插入一条数据。
2. uploadMusicFavorite：通过输入music\_name、music\_id、user\_phone，向数据库favorite表中插入一条数据。
3. uploadMusicShare：通过输入的music\_id、user\_phone、music\_name、share\_date和music\_singer，向数据库中sharesongs表插入一条数据。
4. uploadMusicComments：通过输入的user\_phone、music\_id、music\_name、music\_singer、comments、comments\_date和user\_name，向数据库中comments表插入一条数据。

包含一种修改方法：

1. UPDATEMUSIC：通过输入music\_name、music\_singer、music\_img和music\_storagepath，修改数据库music表中已存在的内容。

包含三种删除方法分别为：

1. delfavoritesong：用于通过输入的music\_name，user\_phone，删除favorite表中符合条件的数据。
2. DELETEMUSIC：用于通过输入的music\_id，删除music表中music\_id为输入值得数据。
3. DELETEMUSICOFUSERF：用于通过输入的music\_id，删除favorite表中收藏music\_id为输入值的数据。

5.4 Service接口

Service包下包含两个接口，以及接口的实现类。分别为UserService和MusicService，以及UserServiceImpl和MusicServiceImpl。分别用来实现对用户和对音乐的操作。

1. UserService和UserServiceImpl

在UserService接口中包含十个对用户信息操作的方法。在UserServiceImpl中一一实现了这些方法。方法如下：

1. userLogin：通过调用UserMapper中的getUserByuser\_phone方法，实现用户的登录检测功能，同时获取该用户的信息返回到controller中。
2. userRegister：调用UserMapper中的userFind方法，首先查询输入的手机号或邮箱是否和user表中的数据重复，实现重复检测功能。再调用UserMapper中的insertUser，如果数据库user表中受影响行数大于0，则表明注册成功，反之则表明注册失败，最后将判断返回到controller中，实现用户的注册功能。
3. userUpdate：调用UserMapper中updateuser方法，实现user表中用户数据的修改，如果数据库user表中受影响函数大于0，则表明修改成功，反之则表明修改失败，最后将判断返回到controller中，实现用户的修改用户信息功能。
4. userFind：在该方法中，通过调用UserMapper中finduser方法，实现对user表中用户信息的查询功能。
5. userfindstring：通过输入的userphone，调用UserMapper中finduserbystring方法，实现通过用户手机号查询user表中用户信息的功能。
6. allUser：通过输入的user\_Administrator，调用UserMapper中的findAllUser方法，实现对user表中不同类别用户（管理员或普通用户）的信息查询。
7. USERUPDATE：调用UserMapper中的FINDUSER方法，并判断该方法返回的ArrayList是否为空，如果不为空，则调用UserMapper中的AUPDATEUSER方法，实现管理员对用户信息修改的功能。
8. DELETEUSER：调用UserMapper中的DELETEUSER方法，如果数据库user表中受影响行数大于0，则表明删除成功，否则表明删除失败。
9. FINDUSER：调用UserMapper中的FINDUSERBYONE方法，将结果以UserBean的形式返回到controller中。
10. INSERTUSER：调用UserMapper中的FINDUSER2方法，判断返回的ArrayList是否为空，如果为空返回FALSE,如果不为空则调用UserMapper中的INSERTUSER方法，如果数据库user表中受影响行数大于0，则表明添加成功，否则表明添加失败。最后将方法的返回值以Boolean类型返回到controller中。
11. MusicService和MusicServiceImpl

在MusicService接口中包含十九个对音乐信息操作的方法。在MusicServiceImpl中一一实现了这些方法。方法如下：

1. findAllMusicBean：通过调用MusicMapper中的findMusicBeanList方法，查询数据库中music表中的所有数据。
2. findMusicByName：通过输入的music\_name，调用MusicMapper中的findMusicByName方法，查询相应的音乐数据，将查询的判断结果和查询结果放入Map中，返回至controller中。
3. findMusicBySinger：通过输入的music\_singer，调用MusicMapper中的findMusicBySinger方法，查询相应的音乐数据，将查询的判断结果和查询结果放入Map中，返回至controller中。
4. uploadMusic：通过输入的newMusicName和newMusicSinger，通过字符串拼接的方法，得到newSongUrl和newSongImgUrl。然后调用MusicMapper中的uploadMusic方法，更新数据库中music表中的数据，通过music表受影响行数判断数据是否上传成功。
5. AccurateFindMusic：通过输入的music\_name，调用MusicMapper中的accurateFindMusic方法，找到对应的音乐信息，按照jplayer插件的音乐播放格式，选择相应数据重新组装，并存放到jsonObject中，然后将jsonObject添加到jsonArray中，得到播放列表，返回jsonArray到controller中。此方法用于播放主页面中的单首歌曲。
6. accuratefindmusicinformation：通过输入的music\_name,调用MusicMapper中的accurateFindMusic方法，获取相应的音乐数据。
7. findMusicOfPlayList：通过输入的user\_phone，调用MusicMapper中的findMusicByUserPhoneOfFavorite方法，获取用户的收藏列表，对找到的音乐数据，按照jplayer插件的音乐播放格式，选择相应的数据重新组装，并存放到jsonObject中，然后将jsonObject添加到jsonArray中，得到播放列表，返回jsonArray到controller中。此方法用于播放用户收藏列表中的歌曲。
8. findMusicOfPlayListinformation：用于通过输入的user\_phone，调用MusicMapper中的findMusicByUserPhoneOfFavorite方法，获取用户的收藏列表中的音乐信息。
9. findONEMusic：通过输入的music\_name，调用MusicMapper中的accurateFindMusic方法获取第一首歌歌曲信息，存放到MusicBean中。
10. delFavoritesong：通过输入的user\_phone和music\_name，调用MusicMapper中的delfavoritesong方法，删除favorite表中相应的数据。
11. uploadFavoritesong：通过输入的FavoriteSongs对象，调用MusicMapper中的uploadMusicFavorite方法，实现在favorite表中插入一条数据。
12. UPDATEMUSIC：通过输入的newMusicName、newMusicSinger和music\_id，调用MusicMapper中的UPDATEMUSIC方法，对music表中相应的数据进行修改。
13. DELETEMUSIC：通过输入的music\_id，调用MusicMapper中的DELETEMUSICOFUSERF方法，删除favorite表中相应的数据。
14. FINDMUSIC：通过输入的music\_name，调用MusicMapper中的FINDMUSICBYNAME方法，获取相应的音乐数据。
15. FINDMUSICOFID：通过输入的music\_id，调用MusicMapper中的FINDMUSICBYID方法，查找music表中相应的数据。
16. shareSongs：通过输入的ShareSongs对象，调用MusicMapper中的uploadMusicShare方法，在sharesongs表中添加数据。
17. SHARE\_SONGS\_ARRAY\_LIST：通过输入的user\_phone，调用MusicMapper中的SHARE\_SONGS\_ARRAY\_LIST方法，查询sharesongs表中相应的数据。
18. insertComments：通过输入的Comments对象，调用MusicMapper中的uploadMusicComments方法，在comments表中插入一条数据。
19. COMMENTS\_ARRAY\_LIST：通过输入的music\_id，调用MusicMapper中的COMMENTS\_ARRAY\_LIST方法，查询comments表中相应的数据。

5.5 Controller方法

在Controller中，分别实现了对用于、音乐、管理员三大类的基本操作。方法分别如下：

1. UserController

在UserController中，实现了七种对于用户的基本操作，其中包含登录、注册、更新、添加收藏列表、删除收藏列表、分享、评论。具体实现功能如下：

1. 登录功能

在登录功能中，首先通过@PostMapping注释，实现登录功能HttpPost的特定映射。再通过@RequestParam注释，获取前端页面提交上来的user\_phone和user\_pwd。对request和response进行UTF8格式化设置。

在逻辑判断中对传入的user\_phone和user\_pwd进行是判空，若为空则在map中存入3作为数据丢失的标识。

若不为空则调用UserService中的userFind方法，对返回的UserBean进行逻辑判断，若为空则在map中存入4作为用户未注册的标识。

若不为空则调用UserService中的userLogin方法，对返回的UserBean进行逻辑判断，若为空则则在map中存入0作为用户登录账号或密码不正确的标识。

若不为空则判断UserBean中user\_Administrator是否为真，若为真则表明是管理员账号，在map中存入11作为管理员登录的标识，并将UserBean存放到命名为ALoginUser的session中。同时调用UserService中的allUser方法和MusicService中的findAllMusicBean方法，将结果分别存入命名为ALLUSER和AMusicList的session中。

若不为真则表明是用户，在map中存入1作为用户登录的标识，并将UserBean存放到命名为LoginUser的session中。同时调用MusicService中的findAllMusicBean方法，将结果存入命名为MusicList的session中。

1. 注册功能

在注册功能中，首先通过@PostMapping注释实现注册功能HttpPost的特定映射。再通过@RequestParam注释，获取前端页面提交上来的user\_name、user\_phone、user\_email和user\_pwd。对request和response进行UTF8格式化设置。

在逻辑判断中对传入的值进行判空，若为空则在map中存入3作为数据丢失的标识。

若不为空则调用UserService中的userRegister方法，对返回的Boolean类型值进行判断，若为假则在map中存入0作为注册失败的标识。

若为真则将UserBean存放到命名为LoginUser的session中。同时在map中存入1作为注册成功的标识。

1. 更新功能

在注册功能中，首先通过@PostMapping注释实现更新功能HttpPost的特定映射。在通过@RequestParam注释，获取前端页面提交上来的user\_new\_name、user\_new\_pwd、user\_new\_birthday和user\_new\_introducedame。对request和response进行UTF8格式化设置。

在逻辑判断中对传入的值进行判空，若为空则在map中存入2作为数据丢失的标识。

若不为空则调用UserService中的userUpdate方法，对返回的Boolean类型值进行判断，若为假则在map中存入0作为更新失败的标识。

若为真则将UserBean存放到命名为LoginUser的session中。同时map中存入1作为更新成功的标识。

1. 添加收藏列表功能

在添加收藏列表功能中，首先通过@PostMapping注释实现添加收藏列表功能HttpPost的特定映射。再通过@RequestParam注释，获取前端页面提交上来的songname和user\_phone。对request和response进行UTF8格式化设置。

在逻辑判断中对传入的值进行判空，若为空则在map中存入3作为数据丢失的标识。

若不为空则调用UserService的userfindstring方法和MusicService的findONEMusic方法和uploadFavoritesong方法，对返回的Boolean类型值进行判断，若为假则在map中存入0作为收藏添加失败的标识。

若为真则在map中存入1作为收藏添加成功的标识，同时存入songname。

1. 删除收藏列表功能

在删除收藏列表功能中，通过@PostMapping注释实现删除收藏列表功能HttpPost的特定映射。再通过@RequestParam注释，获取前端页面提交上来的songname和user\_phone。对request和response进行UTF8格式化设置。

在逻辑判断中对传入的值进行判空，若为空则在map中存入3作为数据丢失的标识。

若不为空则调用MusicService中的delFavoritesong方法，对返回的Boolean类型值进行判断，若为假则在map中存入0作为删除收藏失败的标识。同时在map中存入songname。

若为真则在map中存入1作为删除收藏成功的标识，同时在map中存入songname。

1. 分享功能

在分享功能中，首先通过@PostMapping注释实现分享功能HttpPost的特定映射。再通过@RequestParam注释，获取前端页面提交上来的songid和tips,并对request和response进行UTF8格式化设置。同时调用MusicService中的FINDMUSICOFID方法。

在逻辑判断中对传入的值进行判空，若为空则在map中存入0作为数据丢失的标识。

若不为空则判断tips是否为index，用于区别主页和歌曲信息页面。若是则将MusicBean存放到以sharemusic命名的session中。并在map中存入1作为主页分享的标识。

若tips是share，则获取当前登录用户的session对象和计算机当前的时间，调用MusicService中的shareSongs方法，对返回的Boolean类型值进行判断，若为假则在map中存入0作为分享失败的标识。

若为真则调用MusicService中的SHARE\_SONGS\_ARRAY\_LIST方法，并将方法的返回值存入以sharemusiclist命名的session中，同时在map中存入2作为分享成功的标识。

1. 评论功能

在评论功能中，首先通过@PostMapping注释实现评论功能HttpPost的特定映射。再通过@RequestParam注释，获取前端页面提交上来的comments、music\_name、music\_singer和music\_id,并对request和response进行UTF8格式化设置。

在逻辑判断中对传入的值进行判空，若为空则在map中存入0作为数据丢失的标识。

若不为空则获取当前登录用户的session对象和计算机当前的时间。调用MusicService中的insertComments方法，对返回的Boolean类型值进行判断，若为假则在map中存入2作为评论失败的标识。

若为真则调用MusicService中的COMMENTS\_ARRAY\_LIST方法，并将该方法的返回值存入以commentslist命名的session中。同时在map中存入1作为评论成功的标识。

1. MusicController

在MusicController中，主要实现了四种对于音乐的操作。其中包含通过歌曲名称查询、通过歌手名称查询、动态更新播放列表和获取用户收藏列表。具体功能实现如下：

1. 通过歌曲名称查询功能

在该功能中，首先通过@PostMapping注释实现通过歌曲名称查询功能HttpPost的特定映射。再通过@RequestParam注释，获取前端页面提交上来的music\_name，并对request和response进行UTF8格式化设置。同时获取当前登录用户的session对象。并调用MusicService中的findMusicByName方法，通过循环取出musicByName中的key，判断key是否为TURE，若为假则在map中存入0作为未查询到的标识。

若为真则将key对于的value存放到以searchResult命名的session中。在map中存放1作为查询成功的标识。

1. 通过歌手名称查询功能

在该功能中，首先首先通过@PostMapping注释实现通过歌手名称查询功能HttpPost的特定映射。再通过@RequestParam注释，获取前端页面提交上来的music\_name，并对request和response进行UTF8格式化设置。

1. 动态更新播放列表功能

在该功能中，首先通过@PostMapping注释实动态更新播放列表功能HttpPost的特定映射。再通过@RequestParam注释，获取前端页面提交上来的music\_name，并对request和response进行UTF8格式化设置。

调用MusicService中的AccurateFindMusic方法、accuratefindmusicinformation方法和COMMENTS\_ARRAY\_LIST方法，并将返回结果依次存放到以playMusic、playMusiconeinformation和commentslist命名的session中。在map中存放1作为动态更新成功的标识。

1. 获取用户收藏列表功能

在该功能中，首先通过@PostMapping注释实动态更新播放列表功能HttpPost的特定映射。再通过@RequestParam注释，获取前端页面提交上来的user\_phone，并对request和response进行UTF8格式化设置。

调用MusicService中的findMusicOfPlayList方法，对返回的JSONArray类型值进行判空。若为空则在map中存入0作为获取失败的标识。

若不为空则调用MusicService中的SHARE\_SONGS\_ARRAY\_LIST方法、findMusicOfPlayListinformation，将以上所有方法的结果依次存放在以playMusic、playMusiconeinformation和sharemusiclist命名的session中。并在map中存入1作为查询成功的标识。

1. AdministratorController

在AdministratorController中，实现了六种对于管理员的操作。其中包含用户信息的更新、删除、查询和注册，以及音乐信息的删除和查询。具体实现功能如下：

1. 用户信息更新

在用户信息更新功能中，首先通过@PostMapping注释实现更新功能HttpPost的特定映射。再通过@RequestParam注释，获取前端页面提交上来的user\_name、user\_phone、user\_email、user\_id、user\_introduced和user\_pwd。对request和response进行UTF8格式化设置。

调用UserService中的USERUPDATE方法，对返回的Boolean类型值进行判断，若为假则在map中存入0作为修改失败的标识。

若为真则调用UserService中的allUser方法，更新session中ALLUSER的数据。并在map中存入1作为修改成功的标识。

1. 用户信息删除

在用户信息删除功能中，首先通过@PostMapping注释实现删除功能HttpPost的特定映射。再通过@RequestParam注释，获取前端页面提交上来的user\_id对request和response进行UTF8格式化设置。

调用UserService中的DELETESER方法，对返回的Boolean类型值进行判断，若为假则在map中存入0作为删除失败的标识。

若为真则调用UserService中的allUser方法，更新session中ALLUSER的数据。并在map中存入1作为删除成功的标识。

1. 用户信息查询

在用户信息查询功能中，首先通过@PostMapping注释实现查询功能HttpPost的特定映射。再通过@RequestParam注释，获取前端页面提交上来的user\_phone。对request和response进行UTF8格式化设置。

调用UserService中的FINDUSER方法，对返回的UserBean对象进行判空，若为空则在map中存入0作为查询失败的标识，同时调用UserService中的allUser方法，更新session中ALLUSER的数据。

若不为则将UserBean存放在以ALLUSER命名的session中，以实现更新session中ALLUSER的数据。并在map中存入1作为查询成功的标识。

1. 用户信息注册

在用户信息注册功能中，首先通过@PostMapping注释实现注册功能HttpPost的特定映射。再通过@RequestParam注释，获取前端页面提交上来的user\_phone、user\_email、user\_pwd、user\_name和user\_Administrator。对request和response进行UTF8格式化设置。

调用UserService中的INSERTUSER方法，对返回的Boolean类型值进行判断，若为假则在map中存入0作为注册失败的标识。

若为真则调用UserService中的allUser方法，更新session中ALLUSER的数据。并在map中存入1作为注册成功的标识。

1. 音乐删除

在音乐删除功能中，首先通过@PostMapping注释实现音乐删除功能HttpPost的特定映射。再通过@RequestParam注释，获取前端页面提交上来的music\_id。对request和response进行UTF8格式化设置。

调用MusicService中的DELETEMUSIC方法，对返回的Boolean类型值进行判断，若为假则在map中存入0作为音乐删除失败的标识。

若为真则调用MusicService中的findAllMusicBean方法，更新session中AMusicList的数据。并在map中存入1作为音乐删除成功的标识。

1. 音乐查找

在音乐查找功能中，首先通过@PostMapping注释实现音乐查找功能HttpPost的特定映射。再通过@RequestParam注释，获取前端页面提交上来的music\_name。对request和response进行UTF8格式化设置。

调用MusicService中的FINDMUSIC方法，对返回的ArrayList类型值进行判空，若为空则在map中存入0作为音乐查找失败的标识。

若不为空则将返回值存放在以AMusicList命名的session中，以实现AMusicList的更新。并在map中存入1作为音乐查找成功的标识。

1. FilesController

在FilesController中，实现了四种对于文件的操作。其中包含用户多文件上传、管理员多文件修改、管理员多文件上传和文件下载。具体实现功能如下：

1. 用户多文件上传

在该功能中，首先通过@PostMapping注释实现用户多文件上传功能HttpPost的特定映射。再通过@RequestParam注释，获取前端页面提交上来的newMusicUrl、newMusicImgUrl、newMusicName和newMusicSinger。对request和response进行UTF8格式化设置。

对传入的值进行判空，若为空则重定向到上传页面。

首先将newMusicUrl和newMusicImgURL存放到MultipartFile类型的list数组中。其次通过循环依次判断传入值的后缀名。

若为“.m4a”或“.mp3”，则进行音乐文件上传。设置文件存放的本地路径，设置文件后缀名为“.m4a”，通过transferTo将内存中的音乐数据写入到磁盘中，实现音乐文件上传。

若为“.jpg”或“.png”，则进行图片文件上传。设置文件存放的本地路径，设置文件后缀名为“.jpg”，通过transferTo将内存中的图片数据写入到磁盘中，实现图片文件上传。

最后通过调用MusicService中的uploadMusic方法，对返回的Boolean类型值进行判断，若为假则重定向到用户文件上传页面，表明文件上传失败。

若为真则调用MusicService中的findMusicByName方法，更新session中的searchResult，重定向到歌曲查询页面。

1. 管理员多文件修改

在该功能中，首先通过@PostMapping注释实现用户多文件上传功能HttpPost的特定映射。再通过@RequestParam注释，获取前端页面提交上来的newMusicUrl、newMusicImgUrl、newMusicName、newMusicSinger和music\_id。对request和response进行UTF8格式化设置。

首先将newMusicUrl和newMusicImgURL存放到MultipartFile类型的list数组中。其次通过循环依次判断传入值的后缀名。

若为“.m4a”或“.mp3”，则进行音乐文件上传。设置文件存放的本地路径，设置文件后缀名为“.m4a”，通过transferTo将内存中的音乐数据写入到磁盘中，实现音乐文件上传。

若为“.jpg”或“.png”，则进行图片文件上传。设置文件存放的本地路径，设置文件后缀名为“.jpg”，通过transferTo将内存中的图片数据写入到磁盘中，实现图片文件上传。

最后通过调用MusicService中的UPDATEMUSIC方法，对返回的Boolean类型值进行判断，若为假则重定向到管理员文件上传页面，表明文件更新失败。并调用MusicService中的findAllMusicBean方法，更新session中的AMusicList。

若为真则调用MusicService中的findAllMusicBean方法，更新session中的AMusicList。重定向到管理员界面。

1. 管理员多文件上传

在该功能中，首先通过@PostMapping注释实现用户多文件上传功能HttpPost的特定映射。再通过@RequestParam注释，获取前端页面提交上来的newMusicUrl、newMusicImgUrl、newMusicName和newMusicSinger。对request和response进行UTF8格式化设置。

对传入的值进行判空，若为空则重定向到管理员上传页面。

首先将newMusicUrl和newMusicImgURL存放到MultipartFile类型的list数组中。其次通过循环依次判断传入值的后缀名。

若为“.m4a”或“.mp3”，则进行音乐文件上传。设置文件存放的本地路径，设置文件后缀名为“.m4a”，通过transferTo将内存中的音乐数据写入到磁盘中，实现音乐文件上传。

若为“.jpg”或“.png”，则进行图片文件上传。设置文件存放的本地路径，设置文件后缀名为“.jpg”，通过transferTo将内存中的图片数据写入到磁盘中，实现图片文件上传。

最后通过调用MusicService中的uploadMusic方法，对返回的Boolean类型值进行判断，若为假则重定向到管理员文件上传页面，表明文件上传失败。并调用MusicService中的findAllMusicBean方法，更新session中的AMusicList。

若为真则调用MusicService中的findAllMusicBean方法，更新session中的AMusicList。重定向到管理员界面。

1. 文件下载

在文件下载功能中，首先通过@PostMapping注释实现用户多文件上传功能HttpPost的特定映射。再通过@RequestParam注释，获取前端页面提交上来的music\_id。对request和response进行UTF8格式化设置。

调用MusicService中的FINDMUSICOFID获取音乐的信息，同时获取文件的本地路径，并使用UUID重新命名文件的下载名称。使用FileSystemResource访问本地路径下的文件资源。同时利用HTTPHeaders类中的add方法，对默认下载名称和中文编码定义。然后通过ResponseEntity设置HttpStatus的状态码、自定义标头。最后返换资源。

1. LoginController

在LoginController中，实现了一种对于访客的操作。其中包含。具体实现功能如下：

allMusicBean方法首先通过@PostMapping注释实现游客访问功能HttpPost的特定映射。对request和response进行UTF8格式化设置。调用MusicService中的findAllMusicBean方法，设置MusicList。清空session中其他的数据，保留MusicList。最后返回到主页。

1. 测试与调试

在项目进行中，出现了如下问题：jplayer插件不支持访问本地路径、播放列表无法动态更换，前端页面循环输出结果。

解决方法：由于jplayer插件遵守Adobe的安全协议，所以为了实现播放本地音乐歌曲，在本地建立服务器，从而实现访问本地音乐歌曲。

在jplayer的demo中，播放列表是写死的音乐链接，为了实现动态播放，通过查询数据库中music表中的数据，将返回值按照jplayer播放插件的形式重新组合数据，并将组合好的map数据，放入到JSONArray中，通过session传到前端页面中，最后通过Thymeleaf实现播放列表的动态更新。

在前端页面中，由于采用Thymeleaf模板引擎，所以不再使用标准标签库。通过查询Thymeleaf使用手册后，利用“th:each=${} th:if=${}”代替了标准标签库中的循环，实现了循环输出。

1. 总结

经过了三个多月的努力，终于完成了《基于SpringBoot的爱乐在线音乐共享系统的设计与实现》的SpringBoot项目及论文。

在这个过程中，从对SpringBoot的一知半解到实现应用，每一次的尝试和改进都是自身对Java编程的进步。在不断的调试和修改中，我也更加明白了面向对象的编程思想和MVC的编程思路。充分的体会到了解决问题、突破困难所带给我的激励和快乐。

虽然我的作品不是很成熟，在安全等方面还存在欠缺，但是这每一个功能的实现都是我幸勤劳动的结果。看着自己书写的程序完美的运行，内心仍旧是十分的开心。

这次项目的实现也让我明白了一个道理，一分耕耘一分收获，只有投入大量的时间、精力，不断地研究才可能获得意想不到的结果。也让我体会到了专心研究的乐趣。

在项目和论文的完成过程中得到了张树钧的耐心指导，在此表示衷心的感谢。同时也感谢所有帮助、支持我的人！

参考文献

[1]Maxsu.Thymeleaf教程[EB/OL].https://www.yiibai.com/thymeleaf/,2012.

[2]Thymeleaf团队.Thymeleaf教程[EB/OL].https://www.thymeleaf.org/,2017-04-18.

[3]Phillip Webb, Dave Syer, Josh Long, Stéphane Nicoll, Rob Winch, Andy Wilkinson, Marcel Overdijk, Christian Dupuis, Sébastien Deleuze, Michael Simons, Vedran Pavić, Jay Bryant, Madhura Bhave, Eddú Meléndez.Spring Boot Reference Documentation[EB/OL].https://docs.spring.io/spring-boot/docs/2.2.5.RELEASE/reference/html/,2020-02-27.

[4]大橘子e.源码浅析ResponseEntity.ok[EB/OL].https://www.jianshu.com/p/1238bfb29ee1,2018-08-23.

[5]itzilong.springboot-注解-@Repository、@Service、@Controller 和 @Component[EB/OL].https://blog.csdn.net/zhizhuodewo6/article/details/81365516,2018-08-02.

[6]jQuery中文网.jQuery API 中文文档[EB/OL].https://www.jquery123.com/,2020.

**The design and implementation of springBoot-based Philharmonic online music sharing system and the design and implementation of the SpringBoot-based Philharmonic Online Music Sharing System**

Computer and Information Engineering College Yang Shuo 20161104615

Instructor Zhang Shujun

**Abstract：**Music is always accompanied by life, and the integration of science and technology and music is to continuously improve and improve the current music website for the realization of the old music playback, sharing functions, which to a certain extent to achieve the people's needs for a better life, but also promote the development of the music industry. The realization of the online music playback system makes the music playback function not tied to the specific playing device, the music storage medium is not tied to CD disc, tape, videotape and other physical forms, convenient for music playback and sharing anytime, anywhere. Here I will implement the system using the SpringBoot framework, Mybaits, and BootStrap framework designs, and the development environment uses IDEA. The interface is easy to operate to give the user a minimalist experience.

**Keywords：**Music; SpringBoot Frame; Mybaits; BootStrap Frame;