中南大學

CENTRAL SOUTH UNIVERSITY

本科毕业设计(论文)

GRADUATION DESIGN (THESIS)

题　　目:	Android 多媒体在线管理系统的设计
学生姓名:	李钊伟
指导教师:	彭春华
学　　院:	信息科学与工程学院
专业班级:	通信工程 1301 班

本科生院制

2017 年 5 月

Android 多媒体在线管理系统的设计

摘要

本项目是一套包含服务端和客户端的多媒体在线管理系统。服务端供系统管理员进行音视频资源文件的上传、删除、多条件查询与排序，以及超级管理员对普通管理员的权限管理。服务端 Web 应用基于 Java Web 中 Spring MVC 框架、Spring 框架、MyBatis 框架开发，使用 MVC 模式，将视图层(jsp)、控制层(Controller)、业务逻辑层(Service)分离，各层之间接口清晰，耦合度低。Android 客户端可以对服务端存储的音视频文件进行查询，在线播放或下载播放，同时还提供本地音视频文件播放的功能。此外，管理员在客户端通过身份认证后，可以对服务端的音视频资源进行上传、删除等管理操作。Android 客户端开发使用 MVP 模式，将视图层(View)、表现层(Presenrer)、逻辑层(Interactor)分离。本系统操作界面友好，并且易于扩展和维护。论文主要介绍了本项目的详细开发过程、架构设计、重点技术方案、性能优化方法等内容，对构建一个高性能、低耦合的多媒体在线管理系统具有一定的参考和指导意义。

关键词：多媒体　Android　SSM　MVC　MVP

Design of Android multimedia online management system

ABSTRACT

This project is a set of multimedia and online management system including the server and the client. The server is for the system administrator to upload, delete, multi-condition query and sort the audio and video resource file, and the super administrator is able to manage the general administrator's rights. The development of web application in the server is based on the Spring MVC framework , Spring framework and MyBatis framework of Java Web, separating the view layer (jsp), control layer (Controller), business logic layer (Service) by using MVC mode, which clears interface between the layers , and provides low coupling degree. The android client can query, play or download the audio and video files stored in the server, but also can play local audio and video files. In addition, when the administrator in the client is certificated, the audio and video resources in the server can be uploaded or deleted by the administrator. The development of android client used MVP mode to separate the view layer (View), the presentation layer (Presenrer) and the logical layer (Interactor). The system is user-friendly and easy to expand and maintain. This Thesis introduces the detail of developing process, architecture design, key technical scheme and performance optimization method of this project, which has a certain reference and guidance significance for constructing a high performance and low coupling multimedia online management system.

Key words： Multimedia Android SSM MVC MVP

目录

第 1 章　前言

1.1 项目概况

本项目提供了一套多媒体在线管理系统的解决方案。服务端 Web 应用使用 Java 开发，基于 Spring MVC、Spring、MyBatis 等框架构建，采用 MySQL 数据库。Web 前端使用 Bootstrap 和 jQuery 构建。Android 客户端采用 MVP 框架模式进行项目架构。

多媒体管理系统 Web 服务端提供音视频资源文件上传，分页查看，分类查看，模糊搜索，高级搜索、自定义排序，批量删除和管理员权限管理等功能。

多媒体管理系统 Android 客户端对于音频文件，提供本地播放、在线播放(边下边播)、拖拽进度、上一曲、下一曲、暂停、下载、音量调节等功能，对于视频文件，提供本地播放、在线播放(边下边播)、列表播放、全屏播放、拖拽进度、快进、快退、暂停、下载、亮度调节、音量调节等功能，以及管理员权限下的音视频资源文件上传和删除等功能。

1.2 项目背景和意义

1.2.1 项目背景

如今智能手机已经成为人们生活中不可缺少的一部分，移动端应用也成为了人们生活中频繁使用的软件产品。Android 占领着世界上智能手机操作系统的最大市场份额，拥有最多的用户数。对于开发一套系统而言，针对 Android 平台的应用开发需求愈加增加。此外，近两年短视频、云音乐类智能手机软件的如雨后春笋般层出不穷，需要管理的多媒体资源文件数量愈加增加，多媒体资源文件管理系统的开发需求也愈加增加。本项目在这样的需求背景下，设计了一套包含服务端和客户端的多媒体管理系统，方便了对音视频等多媒体资源文件的管理，同时也为类似系统的开发提供了解决方案。

1.2.2 项目意义

（1）通过使用本项目的服务端 Web 管理应用，使管理员能够更加快速、方便地对服务器上的多媒体资源文件进行管理操作，使得多媒体资源文件的管理工作更加系统化、规范化，提高了管理多媒体资源文件的效率。

（2）通过使用本项目的客户端 Android 应用，使用户能够在友好的界面和交互下，对多媒体资源文件进行本地和在线播放，提高了此类多媒体应用在使用过程中的用户体验。

（3）本项目的源代码已开源到 Github 上(https://github.com/Allenzwli/LMediaSystem)。

其解决方案对开发人员开发类似多媒体在线管理系统具有一定的参考和指导意义。

1.3 项目发展前景

随着移动端短视频，新闻资讯等应用的兴起，使得音视频资源文件的在线播放成为移动端的一个热点技术。本项目所提供的解决方案，对于这类应用在开发过程中有着一定的积极作用。在移动互联网的浪潮下，随着音视频类应用的发展以及云服务逐渐成为互联网基础建设，本项目的在线音视频播放和多媒体音视频资源文件管理的需求将会大大增加。

此外，本项目的客户端适用于 Android 平台，Android 是当下市场份额最高的智能手机操作系统，主要用于智能手机设备，由谷歌公司持续开发和维护，在全球有着海量的用户数。

综上，本项目具有广阔的发展前景。

1.4 国内外研究现状

（1）国内应用现状

“快手”移动 APP 是一款国内时下火热短视频应用，日活跃用户数 3000 万。用户可以上传分享短视频。还可以在线观看他人的视频作品并与视频作者互动。其应用内的大量场景需要对服务端的视频资源进行在线播放，后台也需要对海量的视频资源文件进行管理。

暴风影音是北京暴风科技有限公司推出的一款多媒体播放器产品，包含 Windows、Android、iOS 版本。该播放器兼容大多数的视频和音频格式。其软件特点有:通过自动侦测用户的硬件配置;自动匹配相应的解码器、渲染链;自动调整对硬件的支持。它提供和升级了系统对常见绝大多数影音文件和流的支持，包括 ：RealMedia、QuickTime、MPEG2、MPEG4(ASP/AVC)、VP3/6/7．Indeo、FLV 等流行视频格式；AC3/DTS/LPCM/AAC/OGG/MPC/APE/FLAC/TTA/WV 等流行音频格式；3GP/Matroska/MP4/OGM/PMP/XVD 等媒体封装及字幕支持等。配合 Windows Media Player 最新版本可完成当前大多数流行影音文件、流媒体、影碟等的播放而无需其他任何专用软件。暴风影音采用 NSIS 封装，为标准的 Windows 安装程序，特点是单文件多语种(简体中文 + 英文)，具有稳定灵活的安装、卸载、维护和修复功能，并对集成的

解码器组合进行了尽可能的优化和兼容性调整，适合普通的大多数以多媒体欣赏或简单制作为主要使用需求的用户。

（2）国外应用现状

“ YouTube”是 Google 的一家子公司，目前是世界上最大的视频网站。为全球成千上万的用户提供视频上传、展示、播放等服务。其视频的在线播放功能是通过 FLASH 技术实现的。

美国 FatTail 公司是提供在线广告优化软件的创新公司。FatTail 在 2007 年的时候与国内一家软件公司阿丁特合作开发了一套多媒体资源管理系统 Admanager。该系统在美国倍受欢迎，FatTail 凭借该软件在美国获得 AlwaysOn OnMedia 100 大奖，获得了不菲的奖金。有了国外的成功案例后，阿丁特后来又根据国内媒体行业的特自研发了一套适合国内媒体资源管理系统 Clifford，并成功上线为用户所使用。

1.5 本文的组织结构

本文主要介绍了包含服务端和客户端的多媒体在线管理系统的设计与实现。论文的组织结构安排如下：

第 1 章，绪论，主要介绍了项目的概述、背景、意义、发展前景和国内外研究现状。

第 2 章，开发与运行环境，主要介绍了项目在开发和运行过程中所使用的技术工具和知识。

第 3 章，需求分析，主要介绍了根据毕业设计任务书所分析出来的具体详细的项目需求。

第 4 章，概要设计，主要介绍了系统设计的各个功能子模块、项目架构方案、系统流程图、系统部署图。

第 5 章，详细设计，主要介绍了系统数据库的设计、开发过程中所使用的各个技术点剖析、各子模块的详细实现方案。

第 6 章，运行与测试，主要展示了系统运行后的效果、以及进行单元测试的方法。

第 7 章，性能优化，主要介绍了开发过程中，为了提高系统性能，所使用的优化方法以及可以继续扩展的优化方法。

第 8 章，总结与展望，主要总结了整个项目的开发过程、并介绍了项目中可以优化实现的功能模块。

第 2 章　开发与运行环境

2.1 Web 应用开发环境

2.1.1 概述

（1）开发工具: IntelliJ IDEA 2017.1

（2）Java 版本: 1.8.121

（3）构建工具: Maven

（4）开发机操作系统: macOS Sierra 10.12.4

2.1.2 Web 开发工具概述

（1）IntelliJ IDEA 概述

Intellij IDEA 是 Jetbrains 公司开发的一款 Java 语言集成开发工具。具有代码智能补全、提示、分析、重构、高亮等特性。支持 Junit 单元测试、Maven 依赖构建、丰富的第三方工具插件、版本控制工具(Git、SVN)等，同时具有美观创新的沉浸式 GUI 设计。IntelliJ IDEA 可以极大地提高开发工作的生产效率。

（2）Maven 概述

Maven 是由 Apache 推出的一个软件项目工具，是一个通过 pom 配置文件管理项目构建、库依赖、继续集成的软件项目基础设施。

本项目 Web 应用利用 Maven 工具主要意在对所使用的 Java Web 三大框架 Spring MVC、Spring、Mybatis 进行持续集成与依赖构建。

2.2 Web 应用运行环境

2.2.1 概述

（1）服务器操作系统: CentOS 6.8

（2）Web 容器: Apache Tomcat 8.0.43

（3）数据库: MySQL(InnoDB 引擎) 5.1.73

（4）Java 版本: 1.8.131

2.2.2 运行环境概述

（1）CentOS

CentOS 是一个由 Red Hat 提供的一个企业级 Linux 发行版，适用于要求高度稳定的服务器[1]。本项目选取 CentOS 6.8 作为服务器操作系统，并将 Web 应用部署在一台已购

买的阿里云 ECS 云主机上。

（2）Apache Tomcat 概述

Apache Tomcat 是一个 Web 应用服务器，具有免费、开源、稳定等特点，被广泛用于 JavaWeb 项目的部署[2]。

（3）MySQL 概述

MySQL 是由瑞典 MySQL AB 公司开发的一个关系型数据库，具有体积小，速度快，开源等特点，广泛应用于中小型 Web 应用系统中[3]。

2.3 Android 应用开发环境

2.3.1 概述

（1）开发工具: Android Studio 2.3

（2）SDK 编译版本: API 25(Android 7.1.1)

（3）兼容性: API 16(Android 4.1)以上

（4）构建工具: Gradle

2.3.2 Android 开发工具概述

（1）Android Studio 概述

Android Studio 是一款针对 Android 应用程序开发的集成开发工具，有 Google 公司推出，可以让开发者以此来开发出适用于载有 Android 操作系统的智能手机、智能电视、智能可穿戴设备的应用程序[4]。Android Studio 具有诸多关于移动应用开发的新特性，解决了 Android 应用开发过程中的一些常见问题，如多分辨率适配，多语言国际化等。开发人员利用 Android Studio 可以极大地提高开发生产效率，从而快速开发出高质量、高稳定性的 Android 应用程序。

（2）Gradle 概述

Gradle 的功能类似 Maven，也是一个项目自动化构建工具。它抛弃了传统基于 XML 的各种繁琐配置，而使用 DSL 语言进行项目配置。本项目中 Android 工程使用 Gradle 进行自动化构建，依赖管理。

2.4 Android 应用运行环境

测试机型:小米手机 2S

测试机操作系统:MIUI 8.7.5.11 开发版(基于 Android 5.0.2 LRX22G)

2.5 小结

本章介绍了整个项目开发前的知识准备和项目开发过程中使用的一些关键工具与技术。包括 InterlliJ IDEA 集成开发工具、Maven 构建工具、服务器 CentOS 操作系统、Tomcat 服务器、MySQL 数据库、Android Studio 集成开发工具、Gradle 构建工具等。

第 3 章 需求分析

3.1 任务概述

要求设计一个基于 Android 的多媒体在线管理系统，在智能终端能实现对多媒体服务器内容的查询、一定权限的增删、下载、播放等功能；多媒体内容可在线播放也可下载播放，具有播放、暂停、停止、上下曲选择、音量调节等功能；多媒体服务器可实现音视频资源的增删、分类查询、用户权限管理等功能。

设计界面要求有良好的操作性，设计要求完成多媒体在线管理系统的编程与调试，能在服务器和 Android 智能终端设备上运行。

3.2 功能需求

本项目功能需求分为两大部分，服务端 Web 应用开发需求和客户端 Android 应用开发需求。

3.2.1 服务端 Web 应用的功能需求

（1）身份认证

系统首先需要对使用者的身份进行登录认证，并在身份认证成功后将管理员的已登录状态进行保存。

（2）快捷操作与库存概览

管理页面首页为快捷操作页，页面显示当前多媒体资源文件库中的音频文件数量和视频文件数量，并可点击快捷跳转进对应的多媒体文件添加上传页面。

（3）音频文件管理

管理员可以查看当前库存中所有的音频资源文件信息，包括但不限于音频文件的歌曲名、歌手名、专辑名、专辑封面图片、歌曲时长、原始文件名、文件 URL、文件大小、上传时间等基本信息。

管理员可以分页查看、高级搜索、模糊搜索、自定义排序，单个或批量删除音频资源文件。

管理员可以浏览选择本地设备中存储的音频文件，并上传。为了能够在上传过程中提供更好的用户体验，系统应当在文件上传过程中展示当前文件上传进度百分比。

音频文件上传完成后，系统自动化解析音频文件中的歌曲名、歌手名、专辑名、专辑封面图片、歌曲时长、文件大小等基本信息。

（4）视频文件管理

管理员可以查看当前库存中所有的视频资源文件信息，包括但不限于视频文件的文件名，文件大小、文件 URL、上传时间等基本信息。

管理员可以分页查看、文件名搜索、自定义排序、单个或批量删除视频资源文件。

管理员可以浏览选择本地设备中存储的视频文件，并上传。为了能够在上传过程中提供更好的用户体验，系统应当在文件上传过程中展示当前文件上传进度百分比。

（5）权限管理

系统管理员角色根据权限不同分为普通管理员和超级管理员。

普通管理员具有音视频资源文件的管理权限，普通管理员的数量可以有多个。

超级管理员还额外拥有新增普通管理员和删除普通管理员的权限，超级管理员的数量只能有一个。

3.2.2 客户端 Android 应用的功能需求

（1）本地音频

列表展示智能手机内存储的本地音频文件，点击列表中某一项后开始播放，具有播放、暂停、上一曲、下一曲、音量调节、歌曲封面展示、当前播放进度展示、拖拽进度条播放等功能。

（2）本地视频

列表展示智能手机内存储的本地视频文件，在列表中展示视频文件的缩略图，点击列表中的某一项后可以小窗播放、切换全屏播放。在全屏播放的过程中，具有音量调节、亮度调节、快进、快退、拖拽进度条播放等功能。

（3）在线音频

列表展示服务器上的音频资源文件，点击列表中的某一项后开始在线播放，在进度条中展示当前缓冲进度。具有播放、暂停、上一曲、下一曲、音量调节、歌曲封面展示、当前进度展示、拖拽进度条播放、歌曲文件下载等功能。

（4）在线视频

列表展示服务器上的视频资源文件，在列表中展示视频文件的缩略图，点击列表中的某一项后可以小窗播放、切换全屏播放。在全屏播放的过冲中，具有音量调节、亮度调节、快进、快退、拖拽进度条播放、视频下载等功能。

（5）管理员操作

在 Android 应用内通过管理员身份认证后，管理员可对服务器上的音视频资源文件进行管理操作。管理员可以单个或批量删除服务器上的音视频资源文件。管理员也可浏览选择智能手机本地的音视频文件并上传至服务器。

3.3 性能需求

（1）系统的响应速度

本多媒体管理系统在操作时，能当能够快速处理以及快速反馈，从而达到实时要求，使管理人员在使用系统的过程中中不会因系统响应过慢而影响工作效率。

（2）系统的可维护性和可扩展性

本多媒体管理系统应该考虑可维护性和可扩展性。比如多媒体资源文件的查询的需求可能会在以后不断的更新和完善，如多条件查询或多条件排序。系统应当保持弹性，提供适当的维护以及扩展方法来应对需求的调整和变化。为了实现这一点，在系统开发过程中，应当遵循面向对象编程的设计原则，如开放封闭原则，单一职责原则，让模块与模块之间面向接口耦合。

（3）系统的操作易用性

本多媒体管理系统是提供给用户使用的，应当要求界面美观，具有生动的交互性，操作方式易懂，简洁，易用。

3.4 安全性需求

本多媒体管理系统区分使用者的角色，对于普通未登录用户，只可查看多媒体文件列表以及对音视频文件进行播放；对于音视频文件资源的增删需要管理员权限；对于管理员账号的增删则需要超级管理员权限。故系统在开发过程中，应当充分考虑使用过程中的身份角色的认证和区分以及权限许可，使得本多媒体管理系统在操作过程中具有较高的安全性。

3.5 兼容性需求

因市面上的 Android 手机系统版本不一，并且国内各大 Android 智能手机设备厂商大多拥有各自根据 Android 定制化的 ROM[5]。本项目中的 Android 客户端应用应当具有良好的兼容性，兼容市面上大多数的 Android 智能手机设备。

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	1.0%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.8%
4.1.x	Jelly Bean	16	3.2%
4.2.x		17	4.6%
4.3		18	1.3%
4.4	KitKat	19	18.8%
5.0	Lollipop	21	8.7%
5.1		22	23.3%
6.0	Marshmallow	23	31.2%
7.0	Nougat	24	6.6%
7.1		25	0.5%

Data collected during a 7-day period ending on May 2, 2017.
Any versions with less than 0.1% distribution are not shown.

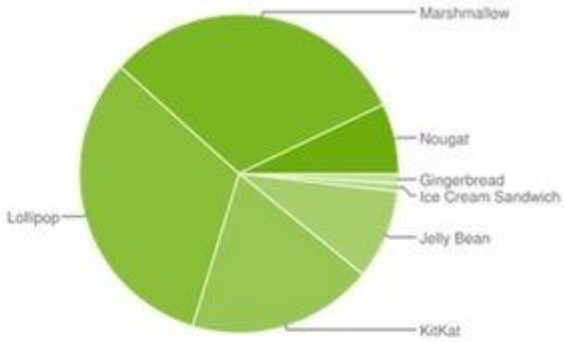


图 3-1　2017 年 5 月 2 日 Android 各版本市场份额统计图

3.6 小结

本章介绍了毕业设计需求的概述、根据概述分析出的详细需求、系统性能需求、安全性需求以及 Android 应用的兼容性需求。

第 4 章　总体设计

4.1 系统模块功能划分

4.1.1 服务端 Web 应用模块划分

（1）权限管理模块

此模块负责对使用者的身份进行认证，即普通管理员还是超级管理员，以及超级管理员增加、删除普通管理员的功能。

（2）音频文件管理模块

此模块负责对服务器上音频文件的上传、删除、查询、解析，以及向 Android 客户端提供对应操作的外部接口。

（3）视频文件管理模块

此模块负责对服务器上视频文件的上传、删除、查询，以及向 Android 客户端提供对应操作的外部接口。

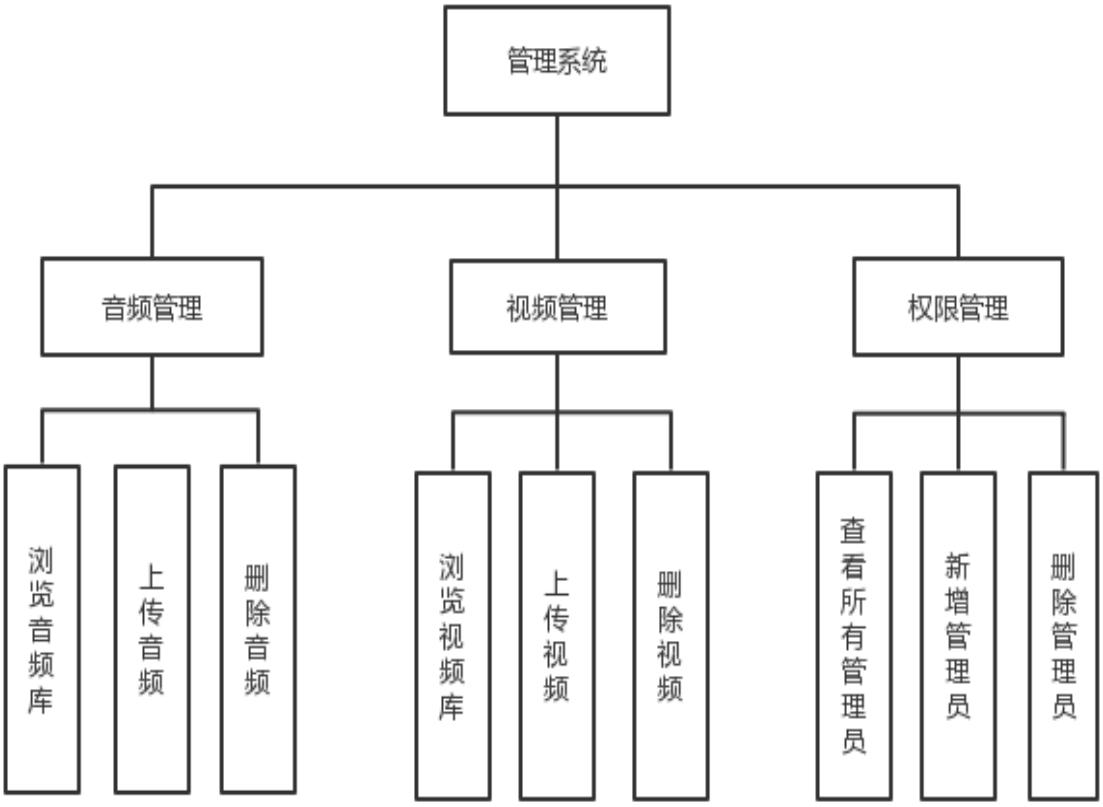


图 4-1　Web 应用模块结构图

4.1.2 客户端 Android 应用模块划分

（1）本地音频获取模块

此模块负责读取 Android 智能手机本地的音频文件并以列表的形式展示出来。

（2）本地视频获取模块

此模块负责读取 Android 智能手机本地的视频文件并以列表的形式展示出来。

（3）在线音频获取模块

此模块负责获取服务端的音频资源文件列表，并以列表的形式展示出来。

（4）在线视频获取模块

此模块负责获取服务端的视频资源文件列表，并以列表的形式展示出来。

（5）音频播放模块

此模块负责播放用户选择的本地或服务端的音频文件，以本地或在线播放的形式。

（6）视频播放模块

此模块负责播放用户选择的本地或者服务端的视频文件，以本地或者在线播放的形式，以及列表小窗播放或者切换全屏播放的形式。

（7）管理员操作模块

此模块负责管理员身份的认证，以及对服务器音视频资源的增加、删除等功能。

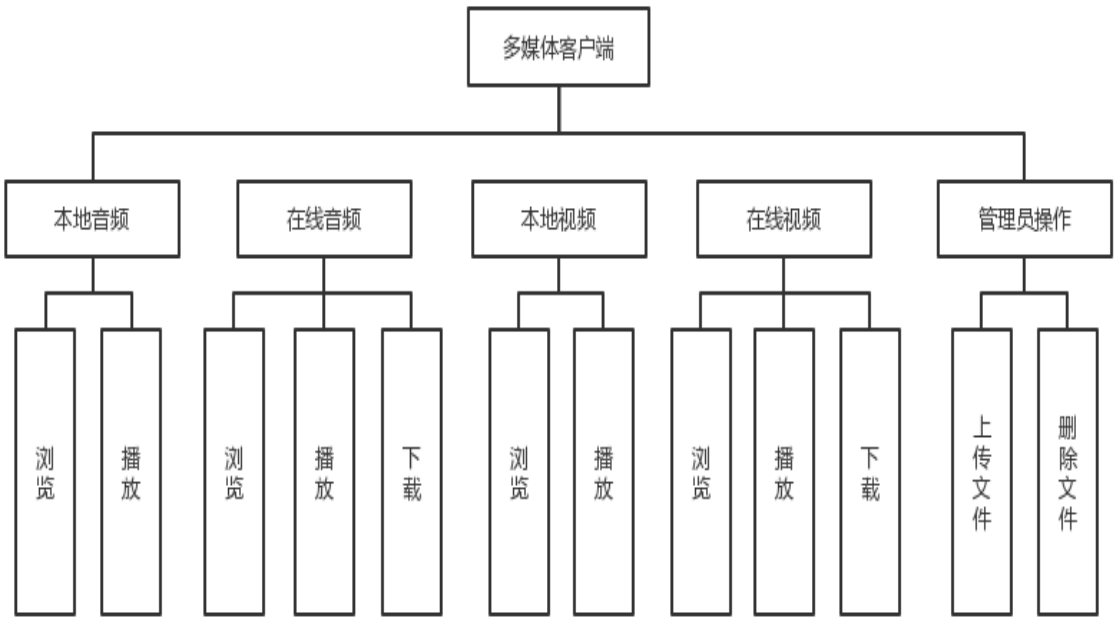


图 4-2　Android 应用模块结构图

4.2 系统架构设计

4.2.1 服务端 Web 应用工程架构

服务端Web应用项目分为视图层(View)、控制器层(Controller)、业务逻辑层(Service)、持久层(Mapper)，各层之间面向接口耦合，针对接口编程，增加系统弹性。

（1）视图层

视图层由众多的 Jsp 前端页面组成，描述了 Web 应用中的用户界面，并使用 JSTL 表达式接收并展示控制器层传递的模型数据。

在本项目中，视图层由 8 个前端 Jsp 页面构成，分别为 admin_add.jsp、admin_manage.jsp、login.jsp、main.jsp、music_add.jsp、music_manage.jsp、video_add.jsp、video_manage.jsp.

（2）控制器层

控制器层由不同的业务控制器 Controller 组成，Controller 接收视图层前端页面发送的 Http 请求，根据请求中不同的 URL 路径和请求过程中传递的参数，来调用对应的业务逻辑层进行处理，获取返回的结果模型数据，并将结果传递给视图层。

在本项目中，控制器层由 3 个 Controller 构成，分别为 AdminController、MusicController、VideoController。

（3）业务逻辑层

业务逻辑层由众多的业务逻辑 Service 组成，按照具体的业务需求，按照不同的业务种类定义不同 Service 接口和方法，并实现。在实现业务逻辑的过程中，调用持久层的 Mapper 对象进行数据库访问操作，如增加、更新、删除、查询等。将 Service 定义成接口，针对接口编程实现相对于直接编写具体的业务逻辑处理代码的好处是，这样可以降低业务逻辑层与控制器层之间的耦合，当项目以后有了新的需求，需要更改业务逻辑时，可以更好的实现代码复用并且不影响其他模块。

在本项目中，业务逻辑层由 4 个 Service 构成，分别为 AdminService、MusicService、VideoService、HomeService.

（4）持久层

持久层负责处理具体的数据库访问操作，如建立数据库连接、查询、增加、删除、更新、断开数据库连接等。将底层的数据库访问方法封装起来，供业务逻辑层调用，将数据库访问的代码和应用程序业务逻辑分离开，降低了业务逻辑和数据库访问之间的耦合，提高了业务逻辑的内聚性。

在本项目中，持久层由 3 个 Mapper 构成，分别为 AdminMapper、MusicMapper、VideoMapper

4.2.2 客户端 Android 应用工程架构

客户端 Android 应用工程分为 View 层、Presentor 层、Interactor 层，各层之间面向接口耦合，针对接口编程，增加了系统弹性。

（1）View 层

View 层负责接收用户输入与应用程序的数据展示，如控件的初始化、按钮的监听事件，加载状态圈的展示和取消。一个 View 应当只持有一个对应 Presentor 的实例，通过接口的方式调用。

在本项目中，定义了 6 个 View 接口：

MainView——由 MainActivity 实现，负责主界面的展示与交互；

MusicView——由 LocalMusicFragment 实现，负责本地音频界面展示与交互；

VideoView——由 LocalVideoFragment 实现，负责本地视频展示与交互；

OnlineMusicView——由 OnlineMusicFragment 实现，负责在线音频界面展示与交互；

OnlineVideoView——由 OnlineVideoFragment 实现，负责在线视频界面展示与交互；

MusicPlayerView——由 MusicPlayerActivity 实现，负责音频播放界面展示与交互。

（2）Presentor 层

Presentor 层负责接收 View 层的输入与调用，并调用 Interactor 层进行对应的数据逻辑处理，将处理的结果再返回给 View 层。一个 Presentor 应当只持有一个 View 实例，但可以持有一个或多个 Interactor 实例，通过接口方式调用。Presentor 起到了关于 View 和 Interacrot 粘合剂的作用。

在本项目中，定义了 4 个 Presentor 接口：

MusicPresentor——由 LocalMusicPresentorImp、OnlineMusicPresentorImp 实现，负责本地音频或在线音频界面交互与数据逻辑的连接；

VideoPresentor——由 LocalVideoPresentorImp、OnlineVideoPresentorImp 实现，负责本地视频或在线视频界面交互与数据逻辑的连接；

MusicPlayerPresentor——由 MusicPlayerPresentorImp 实现，负责音频播放界面交互与数据逻辑的连接；

MainPresentor——由 MainPresentorImp 实现，负责主界面交互与数据逻辑的连接。

（3）Interactor 层

Interactor 层负责具体的数据获取、数据访问和逻辑封装。供 Presentor 来调用。一个 Interactor 对应一个模块的逻辑封装。

在本项目中，定义了 2 个 Interactor 接口：

CommonInteractor——由 LocalMusicInteractorImp、OnlineMusicInteractorImp、LocalVideoInteractorImp、OnlineVideoInteractorImp 实现，负责本地音视频或在线音视频的数据逻辑；

MainInteractor——由 MainInteracorImp 实现，负责主界面的数据逻辑。

4.3 系统流程图

4.3.1 服务端 Web 应用流程图

如图 4-3 所示。

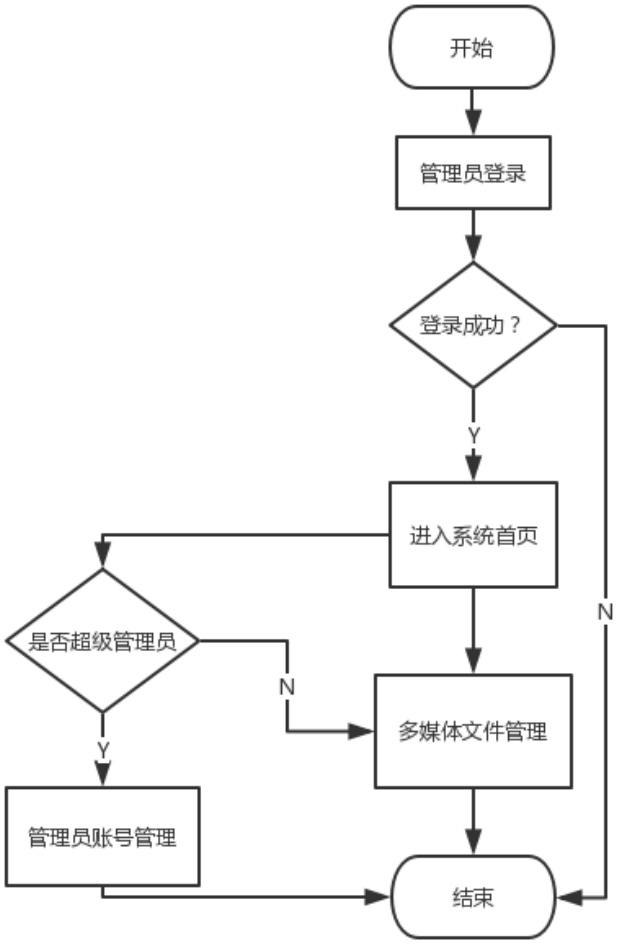


图 4-3 Web 应用流程图

4.3.2 客户端 Android 应用流程图

如图 4-4 所示。

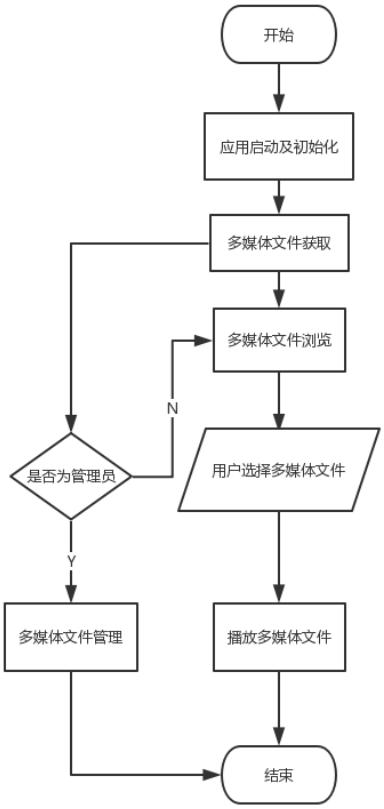


图 4-4　Android 应用流程图

4.4 系统部署图

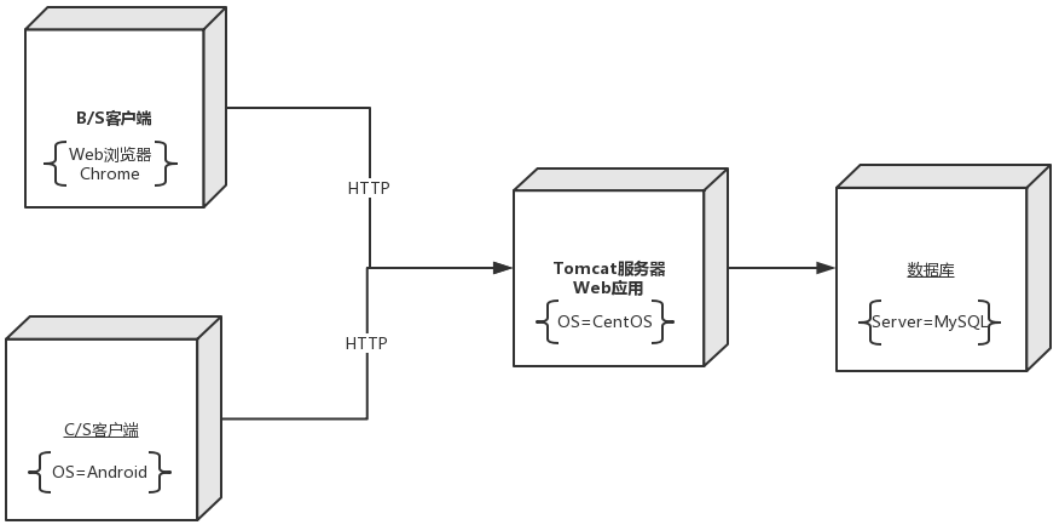


图 4-5　系统部署图

4.5 小结

本章介绍了系统的总体设计，包括系统子模块功能划分、系统项目架构、系统操作流程图、系统部署图。

第 5 章　详细设计

5.1 数据库设计

5.1.1 表设计

本系统数据库包含 3 张表，分别为管理员表 admins,音频文件信息表 songs,视频文件信息表 videos

（1）管理员表(admins)的设计

admins 表用来存储系统中的管理员账号信息，字段包含 id、用户名、加密后的密码、昵称、Token 令牌、创建时间，是否为超级管理员。

表 5-1 管理员信息表(admins)设计

字段名	数据类型	是否允许为空	说明	描述
id	INT	NOT NULL	主键；自增	唯一 ID
account	VARCHAR	NOT NULL		用户名
encypt_password	VARCHAR	NOT NULL		密文密码
nick_name	VARCHAR	NULL		昵称
token	VARCHAR	NOT NULL		Token 令牌
create_time	TIMESTAMP	NOT NULL	默认插入时间	创建时间
is_supper	INT	NOT NULL	默认值为 0	鉴超级管理员

（2）音频文件信息表(songs)的设计

songs 表用来存储管理员上传至服务器中的音频资源文件信息，字段包含 id、歌曲名、歌手名、专辑名、原始文件名、歌曲时长、专辑图片 URL、文件大小、文件 URL、上传时间、上传者的管理员 ID。

表 5-2 音频信息表(songs)设计

字段名	数据类型	是否允许为空	说明	描述
id	INT	NOT NULL	主键；自增	唯一 ID
song_name	VARCHAR	NULL		歌曲名
artist	VARCHAR	NULL		歌手名
album	VARCHAR	NULL		专辑名
file_name	VARCHAR	NOT NULL		原始文件名
duration	BIGINT	NULL		歌曲时长
picture_url	VARCHAR	NULL		专辑图片 URL
upload_time	TIMESTAMP	NOT NULL	默认插入时间	上传时间
admin_id	INT	NOT NULL	外键	管理员 ID
file_url	VARCHAR	NOT NULL		文件 URL
file_size	BIGINT	NOT NULL		文件大小

（3）视频文件信息表(videos)的设计

videos 表用来存储管理员上传至服务器上的视频文件信息，字段包含 ID、视频文件名、文件 URL、文件大小、上传时间

表 5-3 视频信息表(videos)设计

字段名	数据类型	是否允许为空	说明	描述
id	INT	NOT NULL	主键；自增	唯一 ID
video_name	VARCHAR	NOT NULL		视频文件名
file_url	VARCHAR	NOT NULL		文件 URL
file_size	BIGINT	NOT NULL		文件大小
upload_time	TIMESTAMP	NOT NULL	默认插入时间	上传时间
admin_id	INT	NOT NULL	外键	管理员 ID

5.1.2 数据库 E-R 图

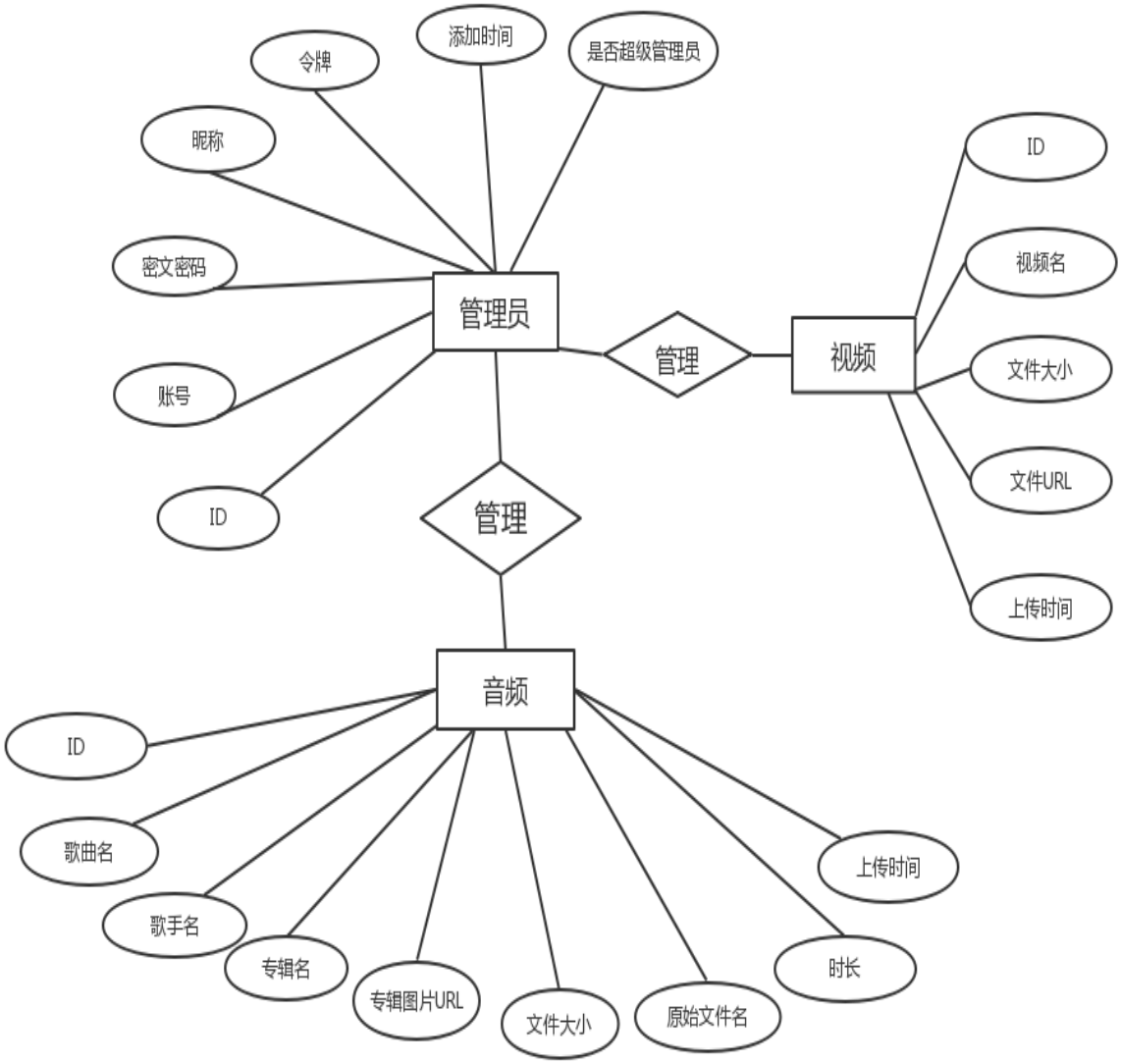


图 5-1 数据库 E-R 图

5.2 服务端 Web 应用开发详细设计

5.2.1 概述

服务端 Web 应用基于 Java Web 开发，使用 MVC 框架模式，自顶向下分为视图层、控制层、业务逻辑层、持久层。使用 Bootstrap 和 jQuery 开发视图层 jsp 前端页面，使用 Spring MVC 框架开发控制层，使用 Spring 框架进行依赖注入、控制反转和事务处理，使用 MyBatis 框架开发持久层。通过 Maven 进行依赖库管理以及持续集成。

5.2.2 主要技术方案与要点

（1）MVC 框架模式概述

MVC 模式全名是 Model(模型)-View(视图)-Controller(控制器)，是一种常用的软件开发框架模式，通常用于 Web 应用开发和 iOS 应用开发中[6]。通过使用这种框架模式，在开发工程中实现封装与分层，将模型数据、视图界面、业务逻辑分离，来降低耦合度，提高系统可扩展性和弹性。如果以后有某一业务需求改变时，只需修改或扩展对应层中代码，而不必修改其他层的代码，提高了代码的可重用性。

Model 负责处理应用中的数据逻辑的部分；

View 负责处理应用中用户界面展示的部分；

Controller 负责获取用户输入，并调用 Model 模型和 View 视图，完成用户请求。

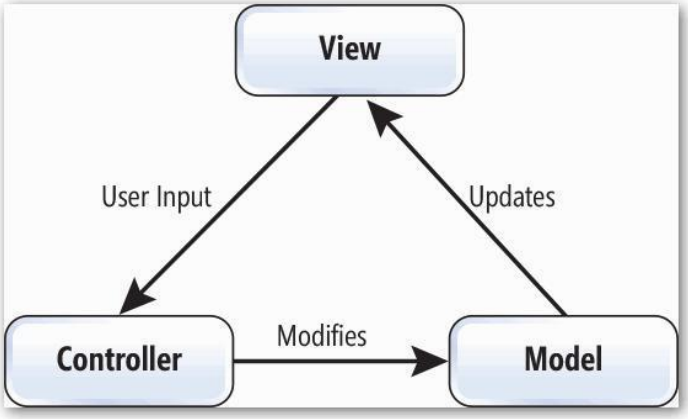


图 5-2　MVC 模式结构图

优点:

1）降低系统内部模块间的耦合；

2）提高代码可复用性和可维护性；

3）有助于代码规范化管理；

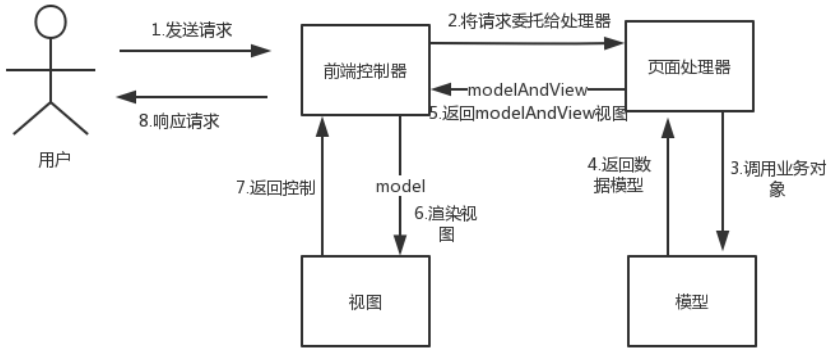
4）提高系统弹性，以更好地应对需求变更。

（2）SSM 框架集概述

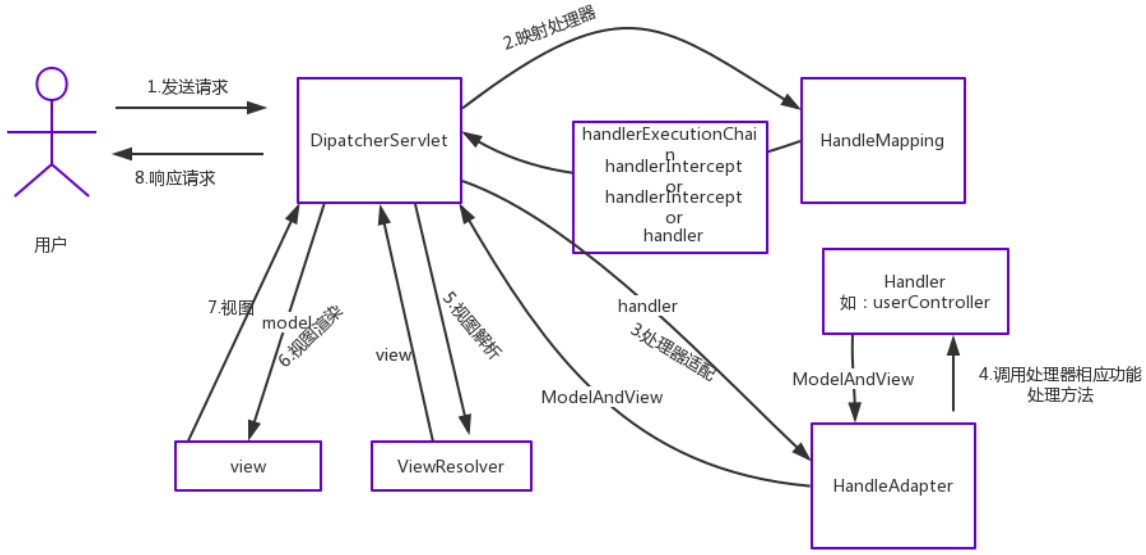
SSM 框架集即指 Spring MVC、Spring、MyBatis 三个框架，是当下最为流行的 Java Web 应用开发框架。

1）Spring MVC 作用在控制层，基于注解或者 XML 配置文件，用户可以创建自己的 Controller 控制器，并通过配置每个 Controller 控制器的 RequestMapping 来拦截对应 URL 的用户请求，调用不同的 Service 业务逻辑组件，并通过 Spring MVC 框架提供的 ModelAndView 对象将结果返回，并通过 Spring MVC 框架内的 ViewResolver 视图解析器来进行页面数据渲染[7]。

Spring MVC 内部由前端控制器、映射处理器、适配处理器、视图解析器等组件构成，具体的内部结构关系与请求处理过程如图 5-3。



springmvc web 请求处理流程



springmvc架构图

图 5-3　Spring MVC 框架内部结构图

2）Spring 框架

Sping 是一个轻量级的 IOC(控制反转)、DI(依赖注入)和 AOP(面向切面编程)的容器框架[8]。

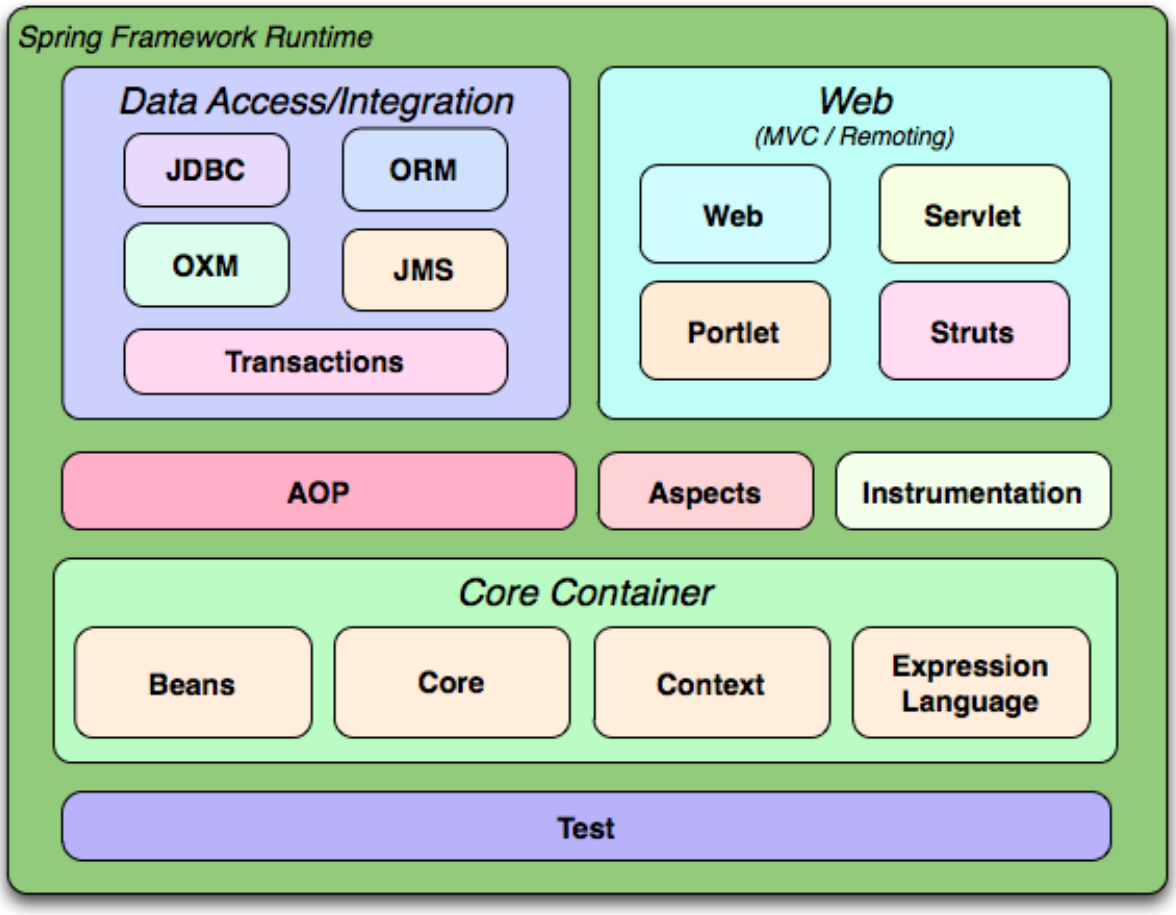


图 5-4　Spring 框架功能结构图

通过使用 Spring 的 IOC 控制反转功能，可以更加容易的组织对象之间的结构关系，降低耦合，以应对项目需求在以后的变更与扩展。

3）MyBatis 框架

MyBatis 是一个作用在持久层的具有 SQL 查询、存储过程和 ORM 高级映射的框架**错误!未找到引用源。**。

MyBatis 使用 XML 配置文件或注解来配置 SQL 语句和结果集映射，通过定义不同的 Mapper 接口和对应的 XML 配置文件，将 Java 的 POJO 对象和数据库表中的一条记录建立映射关系。此外，MyBatis 使用数据库连接池来管理数据库连接，由于创建一个数据库连接是一个耗时，耗资源的过程，使用数据库连接池的方式能够减少这一过程，提高系统查询性能。

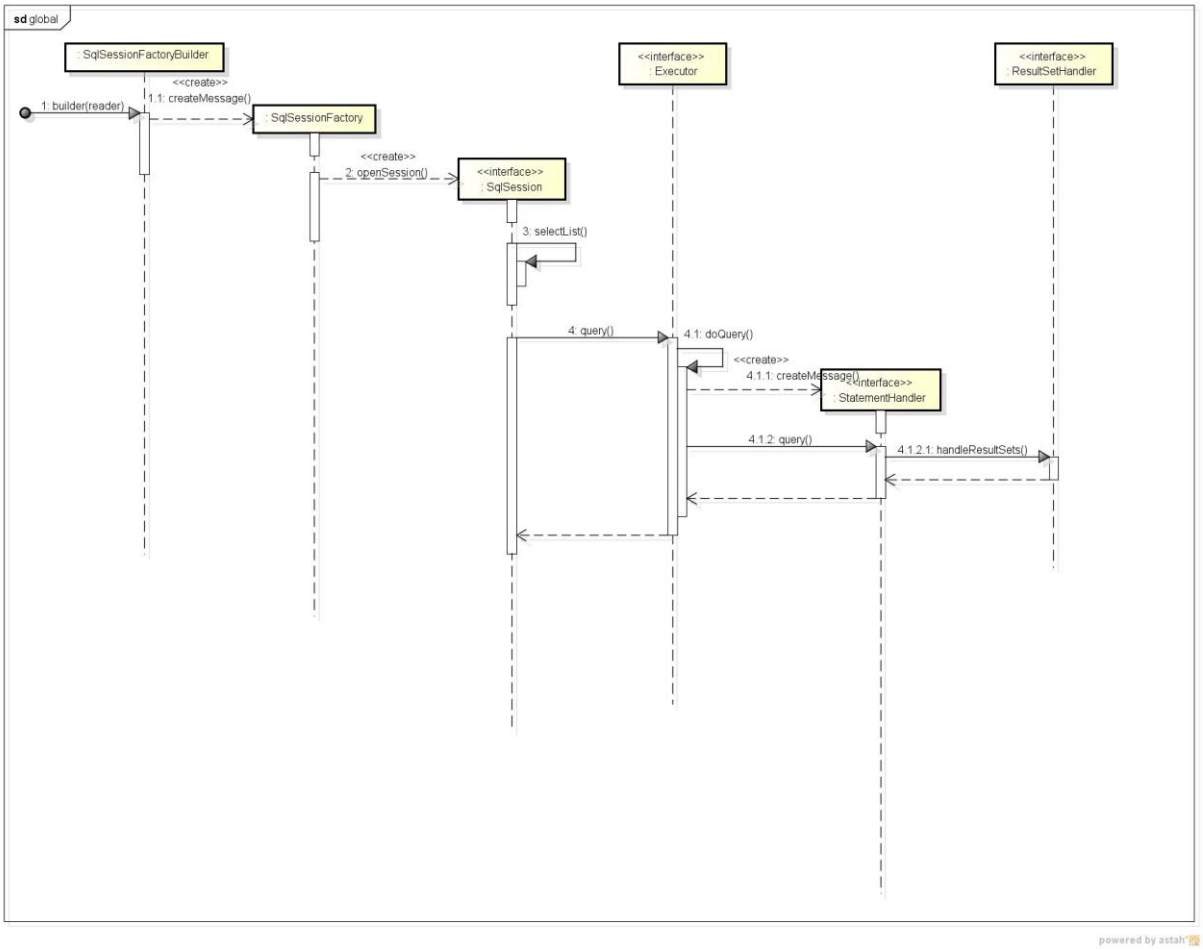


图 5-5　MyBatis 框架整体流程

（2）Bootstrap 概述

Bootstrap 是一个用于前端开发的简洁、直观、功能丰富的工具[9]。可以让 Web 前端开发变得更简单、高效。它是基于 HTML5、CSS3 的，提供了丰富的 Web 前端页面常用组件，如按钮、菜单、导航、对话框等。根据这些组件，能够使开发人员快速地开发出界面美观、功能完备的前端页面。其 CSS 是由 Less 编写，具有一定的动态性，可以让前端页面适配不同的分辨率设备，以达到更好的使用体验。

在本项目中，前端页面的组件元素基于 Bootstrap 创建，同时使页面能够适配 Android 移动端。

（3）jQuery 概述

jQuery 是一个流行已久的高效、快速的跨浏览器 Javascript 框架[11]，它可以优化 HTML 文档的访问和操作、页面组件元素的事件处理、炫酷的动画设计和 Ajax 前后台数据通信交互。

在本项目中，使用 jQuery 进行 Ajax 前后台通信和逻辑处理、表单提交时的参数判断，后台传递的 json 数据解析等。

（4）配置 Maven 项目构建依赖

配置项目的 pom.xml 文件，定义项目需要持续集成的外部依赖框架版本号，添加框架依赖。

```
<!--基本属性定义-->
<properties>
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>

  <!-- spring版本号 -->
  <spring.version>4.2.5.RELEASE</spring.version>

  <!-- mybatis版本号 -->
  <mybatis.version>3.2.8</mybatis.version>

  <!-- mysql驱动版本号 -->
  <mysql-driver.version>5.1.29</mysql-driver.version>

  <!-- log4j日志包版本号 -->
  <slf4j.version>1.7.18</slf4j.version>
  <log4j.version>1.2.17</log4j.version>

  <!--junit版本号-->
  <junit.version>3.8.1</junit.version>

</properties>
```

图 5-6　Maven 配置框架版本号

因 pom.xml 配置文件内容较多，此处只举例 MyBatis 框架的依赖配置，其他框架的依赖配置类似。

```
<!-- 添加mybatis依赖 -->
<dependency>
  <groupId>org.mybatis</groupId>
  <artifactId>mybatis</artifactId>
  <version>${mybatis.version}</version>
</dependency>

<!-- 添加mybatis/spring整合包依赖 -->
<dependency>
  <groupId>org.mybatis</groupId>
  <artifactId>mybatis-spring</artifactId>
  <version>1.2.2</version>
</dependency>
```

图 5-7　Maven 配置 MyBatis 框架依赖

5.2.3 数据模型设计

本项目中需要定义的数据模型主要为 3 个，Admin、Song、Video

（1）Admin 类描述了管理员的基本属性。

表 5-4 管理员数据模型(Admin)设计

属性名	类型	描述
id	int	管理员 ID
account	String	用户名账号
encyptPassword	String	密文密码

nickName	String	昵称
token	String	Token 令牌
createTime	String	创建时间
isSuperAdmin	int	是否超级管理员

（2）Song 类描述了歌曲文件的基本属性。

表 5-5 音频数据模型(Song)设计

属性名	类型	描述
id	int	歌曲 ID
songName	String	歌曲名
artist	String	歌手名
album	String	专辑名
fileName	String	原始文件名
duration	long	歌曲时长
pictureUrl	String	专辑图片 URL
fileSize	long	文件大小
fileUrl	String	文件 URL
uploadTime	String	上传时间
adminId	int	上传者管理员 ID

（3）Video 类描述了视频文件的基本属性

表 5-6 视频数据模型(Video)设计

属性名	类型	描述
id	int	视频 ID
videoName	String	视频文件名
fileSize	long	文件大小
fileUrl	String	文件 URL
uploadTime	String	上传时间
adminId	int	上传者管理员 ID

5.2.4 各模块开发

（1）管理员模块开发

管理员模块的开发主要由以下各组件完成，分别为：

视图层包括 login.jsp 负责身份认证页面的展示，admin_add.jsp 负责添加管理员的操作页面展示，admin_manage.jsp 负责查看所有管理员信息页面的展示。因为添加、删除普通管理员的操作只有超级管理员才拥有权限，故其他页面上点击跳转至 admin_add.jsp 和 admin_manage.jsp 两个页面的<a>标签通过 JSTL 表达式进行了逻辑判断，只有使用者身份为超级管理员时才会显示在页面上。

控制器层的 AdminController 具有增加管理员、删除管理员、获取管理员列表、管

理员登录、管理员登出等功能，通过 Spring MVC 框架拦截对应的请求 URL，并调用业务逻辑层 AdminService 实现。

业务逻辑层的 AdminService 具有注册管理员、删除管理员、获取管理员列表、登录等功能，通过 Spring 框架进行 AdminService 对象的依赖注入，以及 AdminService 对象的生命周期管理，并调用持久层的 AdminMapper 来实现具体逻辑。

持久层的 AdminMapper 具有插入新管理员记录，删除单个管理员记录，查询表内所有管理员记录，查询单个管理员等功能，通过 MyBatis 框架在 XML 文件中配置 SQL 查询语句实现。

（2）音频管理模块开发

音频管理模块的开发主要由以下各组件完成，分别为：

视图层包括 music_add.jsp 音频文件上传页面和 music_manage.jsp 音频文件信息管理页面。

控制器层的 MusicController 具有上传音频文件、单个或批量删除音频文件、分页查询音频文件信息列表、音频文件搜索、获取全部音频文件信息等功能，通过 Spring MVC 框架拦截对应的请求 URL，并调用业务逻辑层的 MusicService 实现具体逻辑。

业务逻辑层的 MusicService 具有保存上传的音频文件、单个或批量删除音频文件、分页获取音频文件信息、高级搜索音频文件、模糊搜索音频文件等功能，通过 Spring 框架进行 MusicService 的依赖注入，以及 MusicService 对象的生命周期管理，并调用持久层 MusicMapper 来实现。

持久层的 MusicMapper 具有新增音频文件信息记录，单个或批量删除音频文件信息记录、多条件查询音频文件、关键字模糊查询音频文件等功能，通过 MyBatis 框架在 XML 文件中配置 SQL 语句实现。

（3）音频文件解析模块

在管理员操作了上传音频文件之后，对于已上传的 MP3 音频文件，需要解析并获取 MP3 文件中所存储的的歌曲名、歌手、专辑、专辑封面图片、歌曲时长等信息。

考虑到对于不同的 MP3 音频文件，其文件中存储待解析信息的 Tag 类型可能不一样，其解析的方法也有不同。为了解除具体的解析算法和业务逻辑的耦合，解析模块使用策略模式进行实现。具体方法如下：

1）定义每种解析算法应当实现的接口 IMusicResolver

```
public interface IMusicResolver {
    String getSongName();
    String getArtist();
    String getAlbum();
    String getPictureUrl();
    long getDuration();
}
```

图 5-8 音频解析器接口设计

2）定义第一种解析算法 ID3V1TagMusicResolverStrategy 类，实现 IMusicResolver 接口，通过 MP3 的 ID3V1 标签解析音频文件信息。

3）定义第二章解析算法 ID3V2TagMusicResolverStragegy 类，实现 IMusicResolver 接口，通过 MP3 的 ID3V2 标签解析音频文件信息。

4）定义音频文件解析器，组合 IMusicResolver 接口类型的成员，提供 public 方法供外部调用以获取 MP3 文件解析后的信息。

```
/**
 * Created by allenzwli
 * 使用策略模式，分离业务逻辑与具体的解析算法，减少耦合
 */
public class MusicResolver {

    private IMusicResolver mMusicResolver;

    public MusicResolver(IMusicResolver i) { mMusicResolver=i; }

    public void setMusicResolver(IMusicResolver i) { mMusicResolver=i; }

    public String performGetSongName() { return mMusicResolver.getSongName(); }

    public String performGetArtist() { return mMusicResolver.getArtist(); }

    public String performGetAlbum() { return mMusicResolver.getAlbum(); }

    public String performGetPictureUrl() { return mMusicResolver.getPictureUrl(); }

    public long performGetDuration() { return mMusicResolver.getDuration(); }

    public Song performGetSongBean() {
        if(mMusicResolver!=null){
            Song song=new Song();
            song.setSongName(mMusicResolver.getSongName());
            song.setArtist(mMusicResolver.getArtist());
            song.setAlbum(mMusicResolver.getAlbum());
            song.setDuration(mMusicResolver.getDuration());
            song.setPictureUrl(mMusicResolver.getPictureUrl());
            return song;
        }
        return null;
    }

}
```

图 5-9 项目中音频解析器设计

5）在 MusicService 的音频文件上传逻辑中调用。

```
MusicResolver musicResolver=new MusicResolver(new ID3V2TagMusicResolverStrategy(file));
if(musicResolver.performGetSongName()==null){
    musicResolver.setMusicResolver(new ID3V1TagMusicResolverStrategy(file));
}
Song song=musicResolver.performGetSongBean();
```

图 5-10 项目中音频解析器调用方法

（4）视频管理模块开发

视频管理模块的开发主要由以下各组件完成，分别为：

视图层包括 video_add.jsp 视频文件上传页面和 video_manage.jsp 视频文件信息管理页面。

控制器层的 VideoController 具有上传视频文件、单个或批量删除视频文件、分页查询视频文件信息列表、视频文件搜索、获取全部视频文件信息等功能，通过 Spring MVC 框架拦截对应的请求 URL，并调用业务逻辑层的 VideoService 实现具体逻辑。

业务逻辑层的 VideoService 具有保存上传的视频文件、单个或批量删除视频文件、分页获取视频文件信息、根据文件名搜索视频文件等功能，通过 Spring 框架进行 VideoService 的依赖注入，以及 VideoService 对象的生命周期管理，并调用持久层 VideoMapper 来实现。

持久层的 VideoMapper 具有新增视频文件信息记录，单个或批量删除视频文件信息记录、根据文件名查询视频文件等功能，通过 MyBatis 框架在 XML 文件中配置 SQL 语句实现。

5.3 客户端 Android 应用开发详细设计

5.3.1 概述

Android 客户端基于 MVP 框架模式开发，在 MainActivity 中使用侧滑菜单，结合自定义的控件 XViewPager，将 5 大模块-本地视频、本地音频、在线视频、在线音频、管理员操作等使用自定义的组件懒加载 LazyFragment 呈现，音频播放器基于 MediaPlayer 和 Service 组件开发，视频播放器基于 IJKPlayer 组件开发。利用 Gradle 进行项目构建依赖管理。

5.3.2 主要技术方案与要点

（1）Android 操作系统概述

Android 是谷歌公司推出的一款开源的操作系统，其主要运行在智能手机、智能可穿戴设备、智能电视等硬件设备上。随着移动互联网的快速发展和智能手机的快速普及、

Android 以其开源免费、美观易用的特点迅速占领了智能手机操作系统市场的大量份额，成为了移动端操作系统的领头人。

1）Android 操作系统架构：

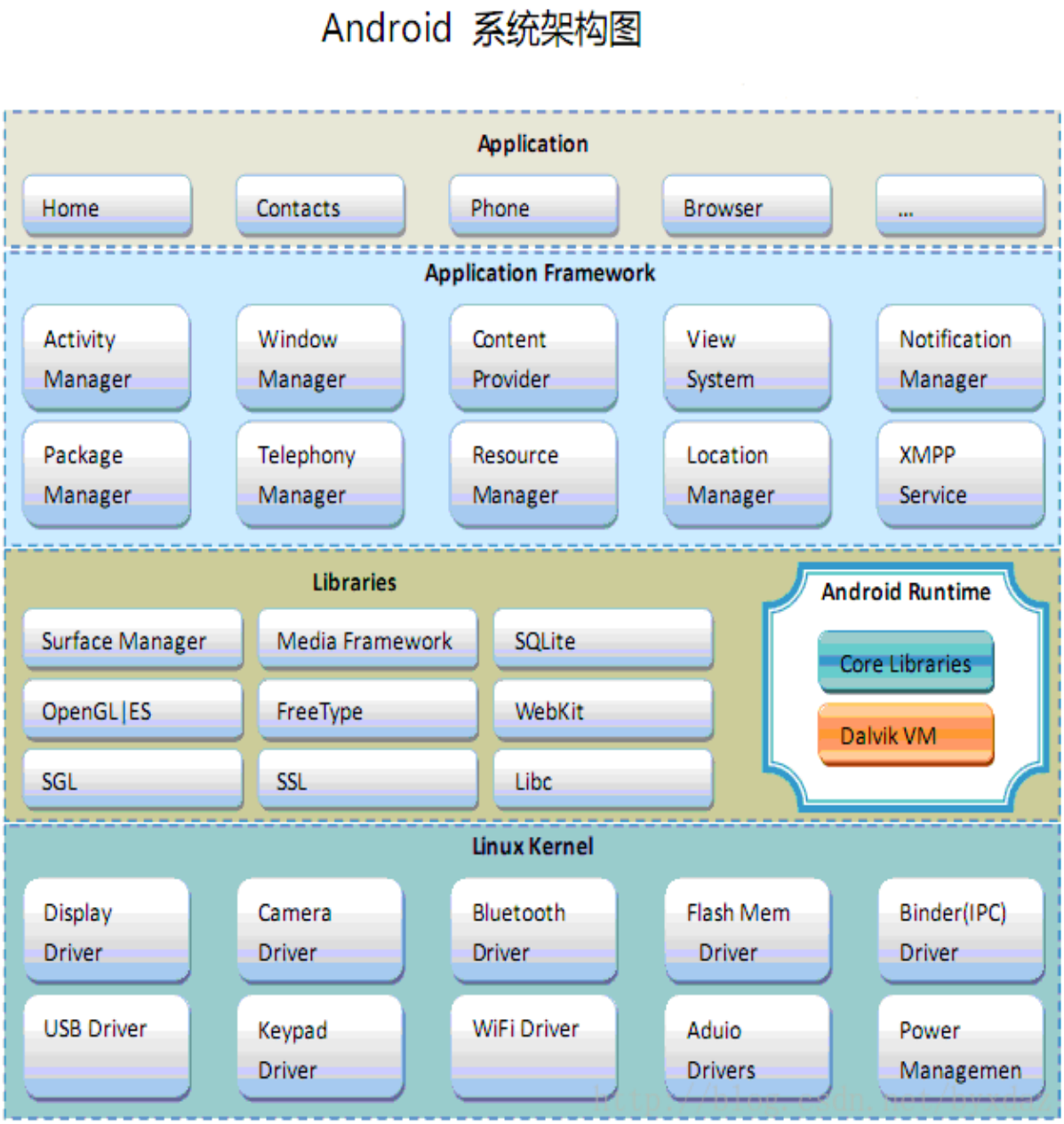


图 5-11 Android 操作系统架构

由上图可以看出，Android 操作系统自顶向下分为 4 层[12]，分别为应用程序层、应用程序框架层、系统运行库层和 Android 运行时、Linux 内核层。各层分别负责各自的职责，提供不同的组件。对于应用程序开发来讲，通常是在应用程序框架层进行开发，根据系统应用程序框架层提供的组件，开发人员可以快速开发出满足自己需求的应用程序。

2）Android 四大组件

Activity,它可以理解为应用程序呈现给用户的界面上的 UI，拥有自己的生命周期，负责处理 UI 控件的初始化和展示，并处理与用户的交互。

Service,它可以理解为一段运行在后台的逻辑程序，拥有自己的生命周期，它没有 UI，通常被用来执行需要长时间、无界面的逻辑操作。在本项目中，音乐播放器的后台播放是通过 Service 组件完成。

BroadcastReceiver 译为即广播接收者，在应用程序中往往有组件与组件之间相互的通信需求，在 Android 中，通信的媒介是 Intent 对象，而 BroadcastReceiver 可以使一个组件注册对某一类事件的广播监听，一旦收到其他组件的发出的 Intent，则会触发 OnReceive 方法，执行需要的逻辑代码。在本项目中，通过 BroadcastReceiver 使得音乐播放的 Service 和播放器界面的 Actvity 进行数据交互，更新进度条、封面等信息。

ContentProvider 类似一个本地的数据库，存储着 Android 手机内的一些本地数据，如短信、通讯录、音频、视频、图像等媒体文件，通过 ContentRosolver 和 Cursor 来进行查询。在本项目中，获取本地的音视频文件信息是通过查询 ContentProvider 实现的[13]。

（2）MVP 框架模式概述

MVP 框架模式的全名是 Model-View-Presenter[14]。如果业务逻辑过于复杂，则会导致 Activity 中 onCreate 方法的逻辑代码过于繁多和复杂，不利于后期迭代与扩展维护。而使用 MVP 框架模式，遵循类的单一职责原则，可以大幅简化 Activity 中的代码，将复杂的业务逻辑交给 View、Presenter、Interactor 三个结合处理。View 的职责是负责应用程序与用户的交互，如展示进度条、更新 UI 等，通常由 Activity 或者 Fragment 担当;Interactor 的职责是负责应用程序数据逻辑的处理，如网络请求、数据库读取等；Presenter 则负责充当胶水般的作用，将 View 和 Interactor 对象连接起来，一个 Presentor 对象通常持有一个 View 对象，一个或多个 Interactor 对象，在 View 中调用 Presenter，Presenter 调用 Inteeractor 进行数据处理，并将结果回调给 View，进行 UI 更新和展示。

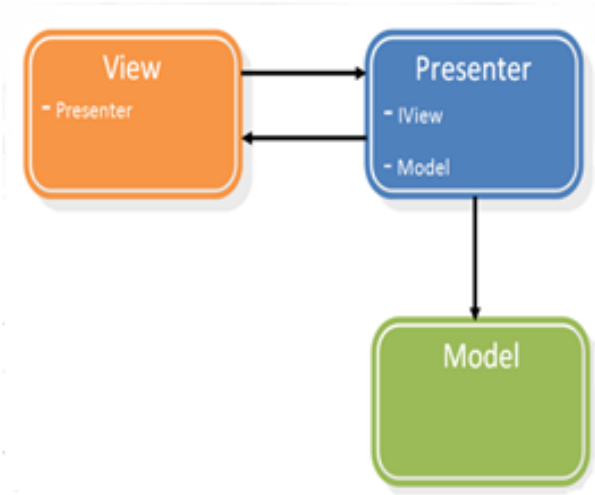


图 5-12 MVP 模式结构图

（3）ButterKnife 注解框架

ButterKnife 是 Square 公司开发的一个 View 绑定和注入框架。在通常的 Android 应用过程中，在 Java 代码侧获取 XML 布局中定义的控件的引用时，需要调用 findViewById 方法。而当一个 Activity 布局中定义的控件较多时，频繁掉使用 findViewById 的重复性不利于开发效率的提高。ButterKnife 框架让开发者可以通过使用注解的方式，在 Java 侧定义 Activity 中的控件时，绑定 XML 布局文件中的控件 ID，建立引用关系。这样类型的注解框架通常是基于 Java 的反射机制实现的，而在运行时使用 Java 的反射机制会对应用程序的性能造成一定的影响，本人也曾经根据反射方案开发过这样一套类似的 View 注解框架。而 ButterKnife 的优势和特点在于它的注解注入方式是通过在编译期生成代码的方式，不会影响应用程序运行时的性能，对应用程序没有副作用，极大地提高了开发人员的开发效率。

在本项目中，音乐播放的 Activity 中定义了很多 View 控件，使用 ButterKnife 通过注解的方式进行 View 绑定。

```
public class MusicPlayerActivity extends BaseActivity implements MusicPlayerView{

    @InjectView(R.id.musics_player_background)
    ImageView mBackgroundImage;

    @InjectView(R.id.musics_player_disc_view)
    CircleImageView mPlayerDiscView;

    @InjectView(R.id.musics_player_play_ctrl_btn)
    ImageView mPlayerCtrlBtn;

    @InjectView(R.id.musics_player_play_next_btn)
    ImageView mPlayerNextBtn;

    @InjectView(R.id.musics_player_play_prev_btn)
    ImageView mPlayerPrevBtn;

    @InjectView(R.id.musics_player_seekbar)
    PlayerSeekBar mPlayerSeekBar;

    @InjectView(R.id.musics_player_name)
    TextView mTitle;

    @InjectView(R.id.musics_player_songer_name)
    TextView mSonger;

    @InjectView(R.id.musics_player_current_time)
    TextView mCurrentTime;

    @InjectView(R.id.musics_player_total_time)
    TextView mTotalTime;

    @InjectView(R.id.volume_seek)
    SeekBar mVolumeSeek;
```

图 5-13 项目中 ButterKnife 注解使用方式

（4）Volley 网络请求框架

Volley 是 2013 年在 Google I/O 开发者大会上推出的一个 Android 网络通信库[15]。在 Android 应用的开发过程中不可避免地经常会遇到与服务端进行 HTTP 协议通信的需求。 Volley 中关于网络请求有三种类,即 StringRequest、JsonRequest、BitmapRequest.分别对应网络请求后返回的结果为 String 字符串、Json 格式的数据、Bitmap 类型的图片数据。同时 Volley 提供了网络请求成功和失败的回调接口，让开发人员可以专注于网络请求的参数设置以及结果处理。

在本项目的在线音频和在线视频模块中，需要向服务端获取当前服务器存储的歌曲或视频信息列表。项目中 Volley 的使用方法如下：

```
@Override
public void getCommonListData() {
    RequestUtil.addRequest(new StringRequest(Request.Method.GET, ApiConstants.ApiUrl.ONLINE_MUSIC_LIST, new Response.Listener<String>() {
        @Override
        public void onResponse(String arg0) {
            List<Song> musicLists = new Gson().fromJson(arg0, new TypeToken<List<Song>>(){}.getType());
            if(musicLists!=null&&musicLists.size()>0){
                mLoadResultListener.onSuccess(0,musicLists);
            }else{
                mLoadResultListener.onError(1,"暂无歌曲");
            }
        }
    }, (arg0) -> {
            mLoadResultListener.onError(2,"请求歌曲列表失败");
    }),getClass().toString());
}
```

图 5-14 项目中 Volley 使用方式

（5）FFmpegMediaMetadataRetriever 解析缩略图

FFmpegMediaMetadataRetriever 是一个对 Android 系统中 MediaMetadataRetriever 类的再实现项目，使用了著名的多媒体处理库 FFmpeg，兼容 ARM, ARMv7, x86 等市场上主流的 CPU 架构智能手机设备，能够在多媒体视频文件中获取帧数据以及媒体元数据，支持设备本地路径的文件, http/https 协议网络地址的文件, 以及 mms, mmsh 和 rtmp 协议地址的文件等。

在本项目中，在视频列表中需要显示视频文件的缩率图作为列表元素的封面图片，通过使用 FFmpegMediaMetadatRetriever，获取视频第 4 秒的帧数据图片 Bitmap。由于这个过程是一个耗时的过程，在开发过程中，应当将这部分逻辑代码放在主线程之外异步完成，从而不影响主线程的 UI 绘制，否则容易触发 ANR。同时，在获取缩略图后应当缓存在本地，供下次使用时直接读取并加载本地缓存好的视频缩略图图片，而不重新获取。项目中的使用方法如图 5-12：

```
public class ThumbUtil {
    public static void saveVideoThumbBitmap(String originalFileUrl,String thumbPicSavePath){
        File file=new File(thumbPicSavePath);
        if(!file.exists()) {
            FFmpegMediaMetadataRetriever fmmr = new FFmpegMediaMetadataRetriever();
            try {
                fmmr.setDataSource(originalFileUrl);
                Bitmap bitmap = fmmr.getFrameAtTime(4000000, FFmpegMediaMetadataRetriever.OPTION_CLOSEST_SYNC);
                if(bitmap!=null){
                    file.createNewFile();
                    FileOutputStream fOut = new FileOutputStream(file);
                    bitmap.compress(Bitmap.CompressFormat.JPEG, 100, fOut);
                    fOut.flush();
                    fOut.close();
                }
            } catch (Exception ex) {
                ex.printStackTrace();
            } finally {
                fmmr.release();
            }
        }

    }
}
```

图 5-15 项目中解析并缓存视频缩略图

（6）Glide 图像异步加载

Glide 是 2014 年 Google I/O 开发者大会上推出的一款轻量、高效、功能丰富的 Android 图片异步加载库。它支持常见的图片格式、集成简单、可配置缓存方式、可进行图像剪裁。

本应用有很多地方需要通过图片网络 URL 来加载服务端存储的网络图片，例如在线音频列表中，在线播放服务端的歌曲时。针对这类需求，在开发过程中选择使用 Glide 库进行网络图像的异步加载。

5.3.3 基类设计

（1）BaseActivity

为了减少 Activity 子类的所做的重复性工作，BaseActivity 重写了 AppCompatActivity 的 onCreate 方法，完成了通用的 setContentView 方法调用，同时定义了两个抽象方法 getContentViewLayoutID 和 initViewsAndEvents 供子类实现，用以设置对应的布局文件 ID 和初始化控件。

（2）BaseLazyFragment

本项目的主要模块均通过 Fragment 展示，但 Android 系统提供的 Fragment 与 ViewPager 结合使用时并没有提供 Fragment 数据状态保持功能，在 ViewPager 切换后，原 Fragment 被销毁，其对应的数据也会丢失，再次通过 ViewPager 切换回来时需要重新创建该 Fragment,在用户侧表现出来的就是每切换一次，Fragment 界面就重新刷新初始化一次，这样频繁的刷新、创建、销毁的过程既不利于应用程序的性能，同时也提供了不友好的用户体验。针对这个问题，本人依据 Fragment 组件的生命周期编写了懒加载的 LazyFragment，在 ViewPager 切换后不会自动销毁，当需要更新 Fragment 里的数据时，用户可以通过下拉等交互操作主动更新，提供了较好的用户体验。

5.3.4 自定义控件

本项目中，根据需求，本人自定义了两个控件

（1）XViewPager

本项目中，主 UI 界面为使用侧滑菜单和 Fragment 结合实现，Android 提供的 ViewPager 控件多用于滑动操作的过程中切换不同 Fragment 页面。而本项目中不希望 ViewPager 进行滑动，而是在点击侧滑菜单的某一项后，让当前 ViewPager 的页面切换为对应的 Fragment。

```
public class XViewPager extends ViewPager {

    private boolean isEnableScroll = true;

    public void setEnableScroll(boolean isEnableScroll) { this.isEnableScroll = isEnableScroll; }

    public XViewPager(Context context) { super(context); }

    public XViewPager(Context context, AttributeSet attrs) { super(context, attrs); }

    @Override
    public boolean onInterceptTouchEvent(MotionEvent ev) {
        if (!isEnableScroll) {
            return false;
        }
        return super.onInterceptTouchEvent(ev);
    }

    @Override
    public boolean onTouchEvent(MotionEvent ev) {
        if (!isEnableScroll) {
            return false;
        }
        return super.onTouchEvent(ev);
    }
}
```

图 5-16 自定义 XViewPager

如图代码所示，自定义 XViewPager 继承自 ViewPager,根据 Android 的 Touch 事件分发机制，重写父类 ViewPager 的 onInterceptTouchEvent 方法和 onTouchEvent 方法，返回 false,表示 Touch 事件至此处理完毕,不需要继续传递。

（2）PlayerSeekBar

在音频播放 Activity 界面中，需要显示当前音频播放的进度，并且可以拖拽播放。Android 提供的默认 SeekBar 控件不够美观，本人模仿市面上的主流音视频 APP 的播放进度条设计，自定义编写了 PlayerSeekBar，继承 SeekBar 并在 onDraw 方法中使用 Canvas 绘图来实现。

```
@Override
protected synchronized void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    if (drawLoading) {
        canvas.save();
        degree = ((int) (degree + 3.0F));
        degree %= 360;
        matrix.reset();
        matrix.postRotate(degree, loading.getWidth() / 2, loading.getHeight() / 2);
        canvas.translate(
                getPaddingLeft() + getThumb().getBounds().left + drawable.getIntrinsicWidth() / 2 - loading.getWidth() / 2 - getThumbOffset(),
                getPaddingTop() + getThumb().getBounds().top + drawable.getIntrinsicHeight() / 2 - loading.getHeight() / 2);
        canvas.drawBitmap(loading, matrix, null);
        canvas.restore();
        invalidate();
    }

}
```

图 5-17 自定义 PlayerSeekBar 核心代码

运行时效果如下：

图 5-18 自定义 PlayerSeekBar 效果

5.3.5 各模块开发

（1）主界面模块开发

主 UI 模块由 MainActivity、MainInteractorImp、MainPresenter 组件构成。

1）MainActivity 实现了 MainView 接口，负责在 Fragment 集合初始化完成后，将其传入 XViewPager 的适配器中，进而传入 XViewPager 对象中。此外，还负责初始侧滑菜单 DrawerLayout、标题 ToolBar、XViewPager，并调用 MainPresenter 进行 Fragment 集合的初始化与对应绑定。

2）MainInteractoImp 实现了 MainInteractor 接口，负责创建 Fragment 集。

3）MainPresenterImp 实现了 MainPresenter 接口，负责调用 MainInteractorImp 进行 Fragment 集合的创建，并回调 MainView。

（2）本地音频获取模块开发

本地音频获取模块由 LocalMusicFragment、LocalMusicInteractorImp、LocalMusicPresenterImp 组件构成。

1）LocalMusicFragment 实现了 MusicView 接口，负责刷新音频列表数据，和点击事件的 UI 处理。

2）LocalMusicInteractorImp 实现了 CommonInteractor 接口，负责从本地查询获取音频文件的数据，通过 ContentResolver 和 Curor 查询 Android 的媒体库 ContentProvier 中的所有音频文件信息。

3）LocalMusicPresenterImp 实现了 MusicPresenter 接口，负责调用 LocalMusicInteractorImp 进行本地音频文件列表的获取和点击事件的数据逻辑处理，并回调 MusicView。

（3）本地视频获取模块开发

本地视频获取模块由 LocalVideoFragment、LocalVideoInteractorImp、LocalVideoPresenterImp 组件构成。

1）LocalVideoFragment 实现了 VideoView 接口，负责刷新视频列表数据，和点击事件的 UI 处理。

2）LocalVideoInteractorImp 实现了 CommonInteractor 接口，负责从本地查询获取视频文件的数据，通过 ContentResolver 和 Curor 查询 Android 的媒体库 ContentProvier 中的所有视频文件信息。

3）LocalVideoPresentreImp 实现了 VideoPresenter 接口，负责调用

LocalVideoInteractorImp 进行本地音频文件列表的获取和点击事件的数据逻辑处理，并回调 VideoView。

（4）在线音频获取模块开发

在线音频获取模块由 OnlineMusicFragment、OnlineMusicInteractorImp、OnlineMusicPresenterImp 组件构成。

在线音频获取模块的开发和本地音频获取模块的开发类似，区别在于 OnlineMusicInteractor 的数据获取是通过 HTTP 网络请求获取服务端返回的音频列表数据。

（5）在线视频获取模块开发

在线视频获取模块由 OnlineVideoFragment、OnlineVideoInteractorImp、OnlineVideoPresenterImp 组件构成。

在线视频获取模块的开发和本地视频获取模块的开发类似，区别在于 OnlineVideoInteractor 的数据获取是通过 HTTP 网络请求获取服务端返回的视频列表数据。

（6）音频播放列表模块开发

音频播放列表模块由 PlayListSingletom 组件构成。

由于在音频播放器的应用体验中，不管是本地还是在线进行音频播放，歌曲的音频播放列表只有一份，故使用单例模式开发音频播放列表。PlayListSingleton 类维护了一个歌曲集合类型的成员变量，用来抽象并描述歌曲播放列表，并提供对外的刷新、情况、单个添加等方法。

```
public class PlayListSingleton {

    private static PlayListSingleton instance;

    private PlayListSingleton() { mSongLists=new ArrayList<Song>(); }

    public static PlayListSingleton getInstance(){
        if(null==instance){
            synchronized (PlayListSingleton.class){
                if(instance==null){
                    instance=new PlayListSingleton();
                }
            }

        }
        return instance;
    }

    private List<Song> mSongLists;
```

图 5-19 播放列表核心代码

（7）音频播放模块开发

音频播放模块使用 MusicPlayer、MediaPlayer、MusicPlayerService、MusicPlayerActivity、MusicPlayerPresenterImp 实现。

1）MusicPlayer 负责管理播放操作指令以及播放状态信息发送，接收 MusicPlayerActivity 的操作指令、如下一曲、上一曲、播放、暂停等，并向 MusicPlayerActivity 发送广播，传递当前的播放进度、歌曲信息。

2）MediaPlayer 负责音频文件的核心播放逻辑。它是 Android 系统提供的用于音频文件播放的组件，支持通过本地文件 URI 或者网络文件 URI 播放本地或者网络的音频文件，并且可以通过实现 OnBufferUpdate 接口，回调缓冲进度。。

3）MusicPlayerService 继承自 Android 系统中的 Service 组件，用于在后台播放音频文件。当用户在关闭了 MusicPlayerActivity 后，应用程序仍然能够继续在后台播放音频文件。

4）MusicPlayerPresenterImp 负责音频播放过程中，Activity 的界面数据更新和 UI 交互逻辑。

5）MusicPlayerActivity 负责音频播放过程中的 UI 展示和界面交互，接收 MusicPlayer 对象传来的广播。根据传递的音频文件播放信息数据调用 MusicPlayerPresenterImp 更新 UI 界面，同时也接收用户在界面的操作输入如、点击下一

曲、上一曲、拖拽进度等，并将操作指令发送广播。

（8）视频播放模块开发

视频播放器模块基于 Bilibili 的开源项目 IJKPlayer 开发。将其作为视频列表中的单元组件，在视频列表适配器中传递视频文件 URL，实现列表小窗播放，切换全屏播放。

（9）管理员操作模块开发

管理员模块主要由 AdminManageFragment 实现，使用 WebView 控件，加载服务端 Web 应用针对移动设备适配的多媒体管理页面，在身份认证成功后，实现本地文件上传、在线资源删除等功能。

5.4 服务端与客户端通信接口设计

Android 客户端与 Web 服务端需要使用 Http 协议进行数据通信，具体接口约定规范如下：

表 5-7 服务端与客户端通信接口设计

接口名	URL	请求方式	返回数据格式
获取音频列表	/musics/list	GET	Json
获取视频列表	/videos/list	GET	Json

5.5 安全性设计

5.5.1 概述

本系统对音视频资源文件的管理需要管理员权限，本系统使用了 Session 验证(用于服务端 Web 应用)和 Token 验证(用于客户端 Android 应用)两种方式。

5.5.2 Session 验证

对于服务端的 Web 应用而言，项目使用 Session 验证机制控制系统访问与操作的安全性[16]。在管理员访问系统首页进行身份认证登录后，如果后台验证用户名和密码成功，即管理员身份认证正确，则后台将此管理员的身份信息放入 Session 中。在之后处理该管理员的操作时，判断 Session 中的管理员信息是否为空，如果为空，则认为管理员的身份认证信息失效，系统跳转至身份认证登录页面，需要重新认证；如果不为空，则认为管理员当前已经是登录态，已拥有音视频管理页面的访问和操作权限，可以正常访问音视频管理系统以及进行页面跳转。对于 Session 中管理员信息失效的条件，既可以通过项目配置文件配置 Session 最大保持时长，也可以通过点击管理系统页面内的登出链接进行账号登出操作，使 Session 失效。

对于客户端的 Android 应用而言，需要管理员身份认证的诸如添加、删除等操作，

是通过 Android 应用内使用本地 WebView 加载管理系统适配移动端的网页来实现，故在网页身份认证登录成功后，其操作安全性的控制方法和服务端 Web 应用所采用的 Session 验证机制相同。

5.5.3 Token 验证

对于客户端的 Android 应用而言，项目使用 Token 验机制控制系统访问与操作的安全性。Token 译为令牌，是由服务端 Web 应用在处理登录操作时，如果管理员身份认证成功，则下发给客户端的一段字符串，作为管理员身份保留在客户端的一个凭证。Android 客户端在收到 Token 令牌后，将其保存在本地，在之后进行需要权限的行为操作时，不必重复发送用户名和密码，只需要带上这个 Token 令牌凭证即可。对于 Token 令牌的值，本系统使用管理员注册时的 JSessionID 作为其 Token 令牌值。

5.6 小结

本章介绍了整个系统的详细设计，包括数据库设计、服务端 Web 应用开发、客户端 Android 应用开发、服务端和客户端通信接口、安全性设计等内容。

第 6 章　运行与测试

6.1 运行结果

6.1.1 服务端 Web 应用运行效果

在毕业设计期间，Web 应用项目部署在我的一台服务器上。

访问 URL: http://112.74.175.94:8080/LMediaServer/

（1）身份认证

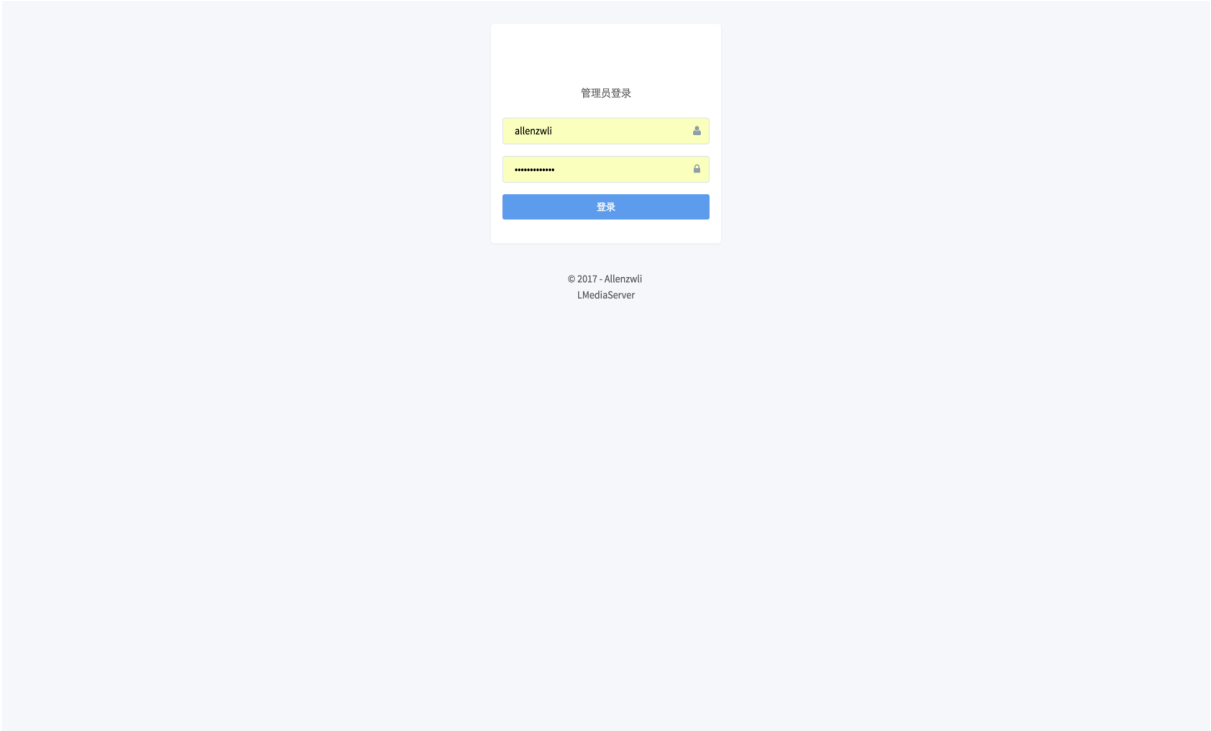


图 6-1 身份认证页面

（2）首页-快捷操作

蓝色色块板展示了当前数据库中的音频文件数量，并且可以点击跳转至增加音频页面。

紫色色块班展示了当前数据中的视频文件数量，并且可以点击跳转至增加视频的页面。

图 6-2 快捷操作首页

（3）音频文件管理

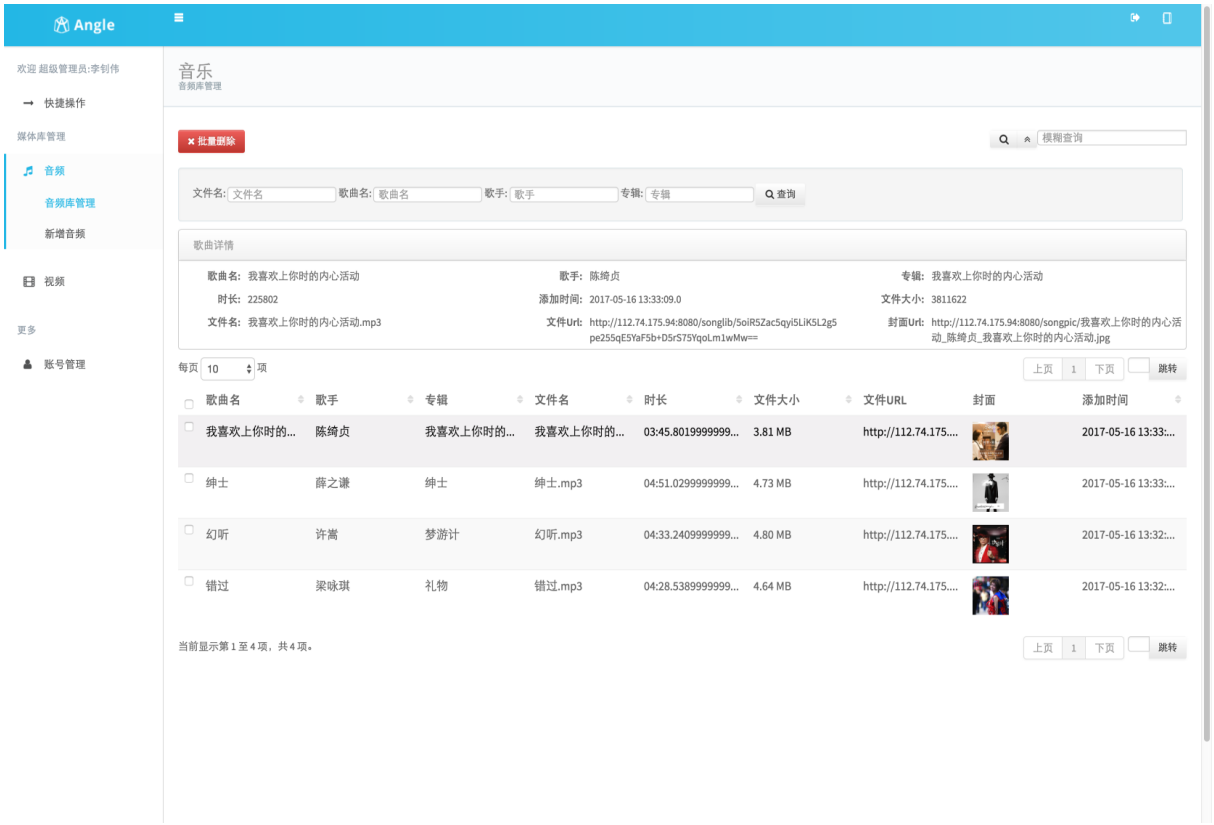


图 6-3 音频管理页

（4）新增音频

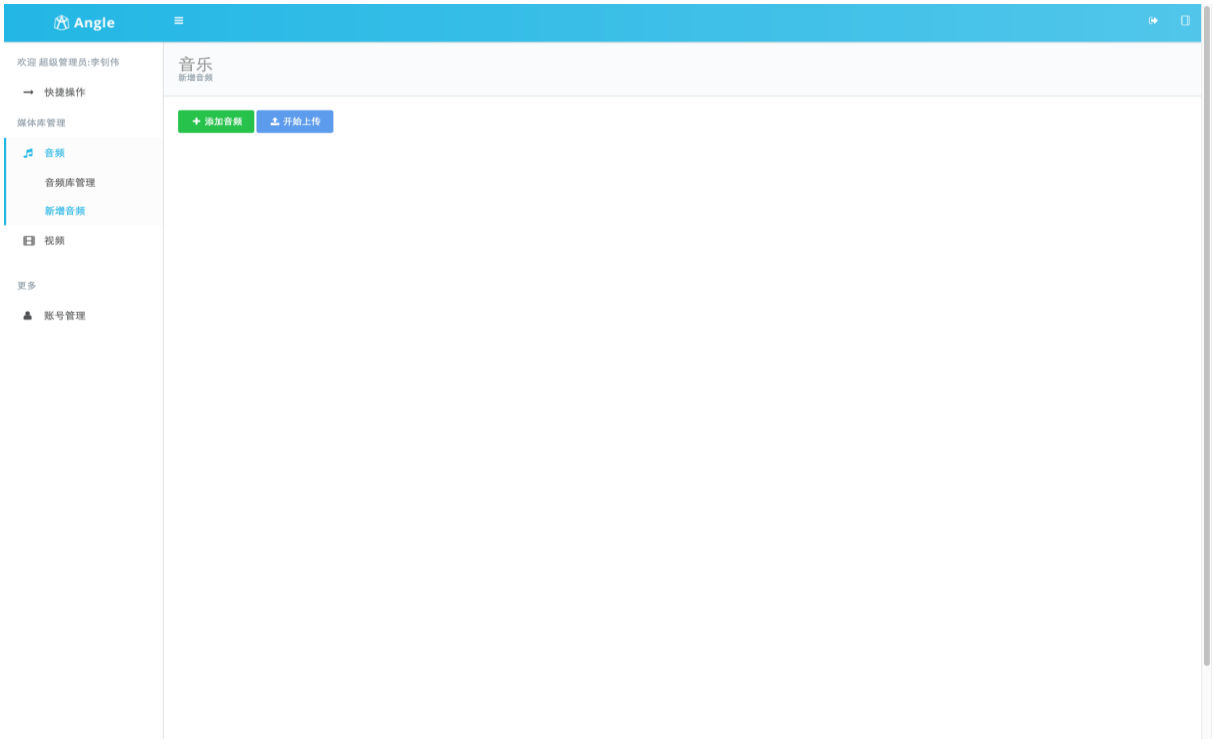


图 6-4 新增音频页

（5）视频文件管理

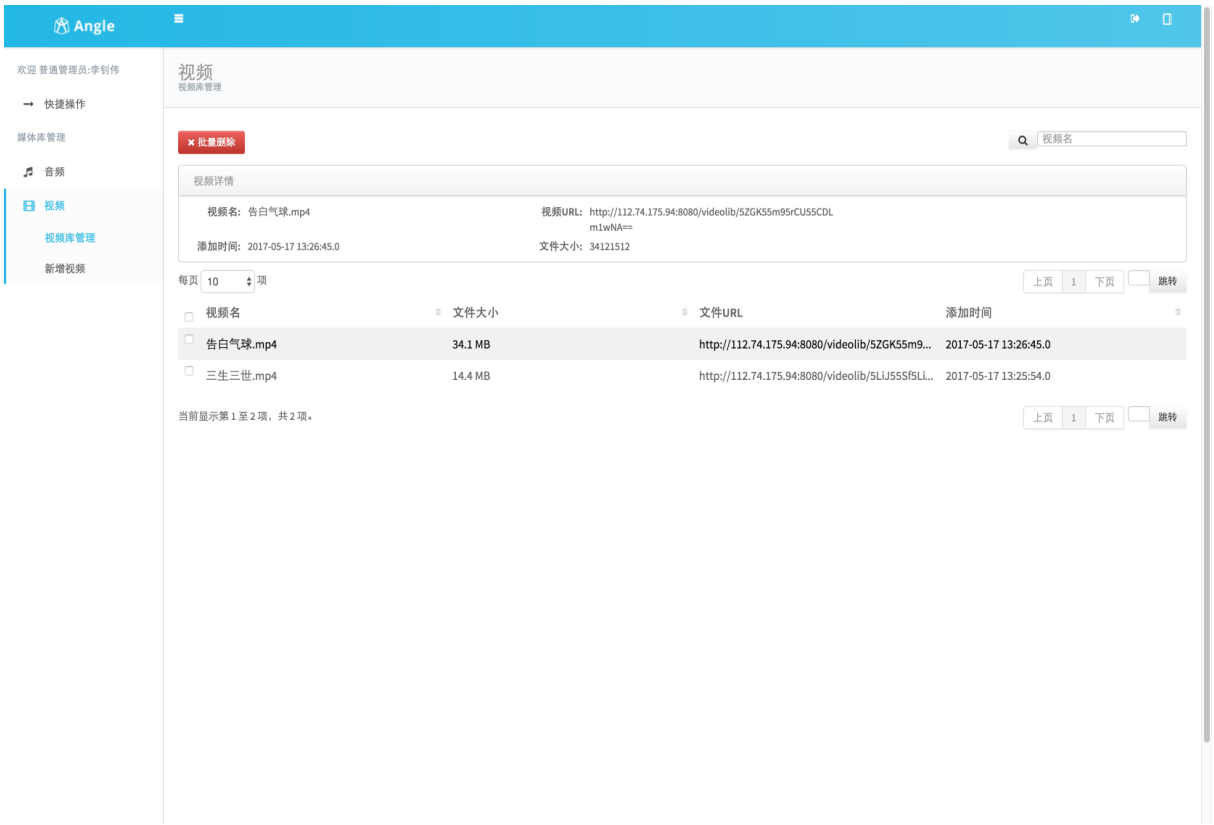


图 6-5 视频管理页

（6）新增视频

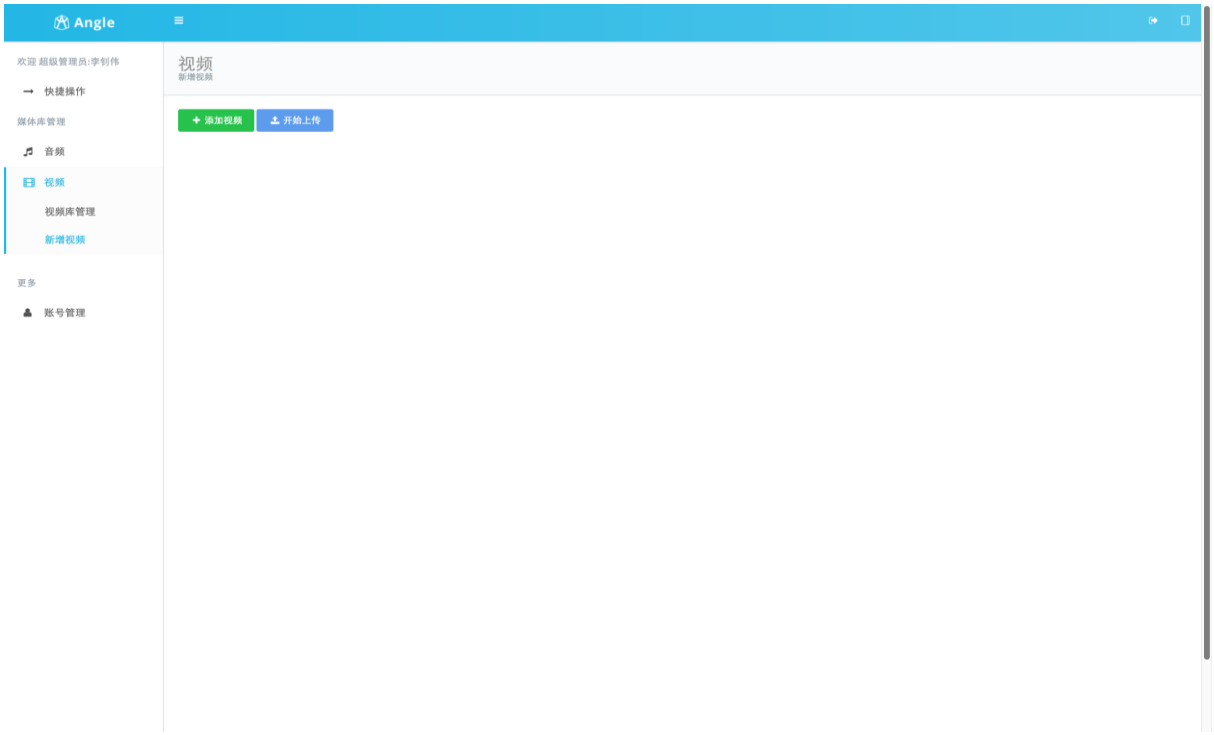


图 6-6 新增视频页

（7）管理员账号管理（超级管理员身份下可见）

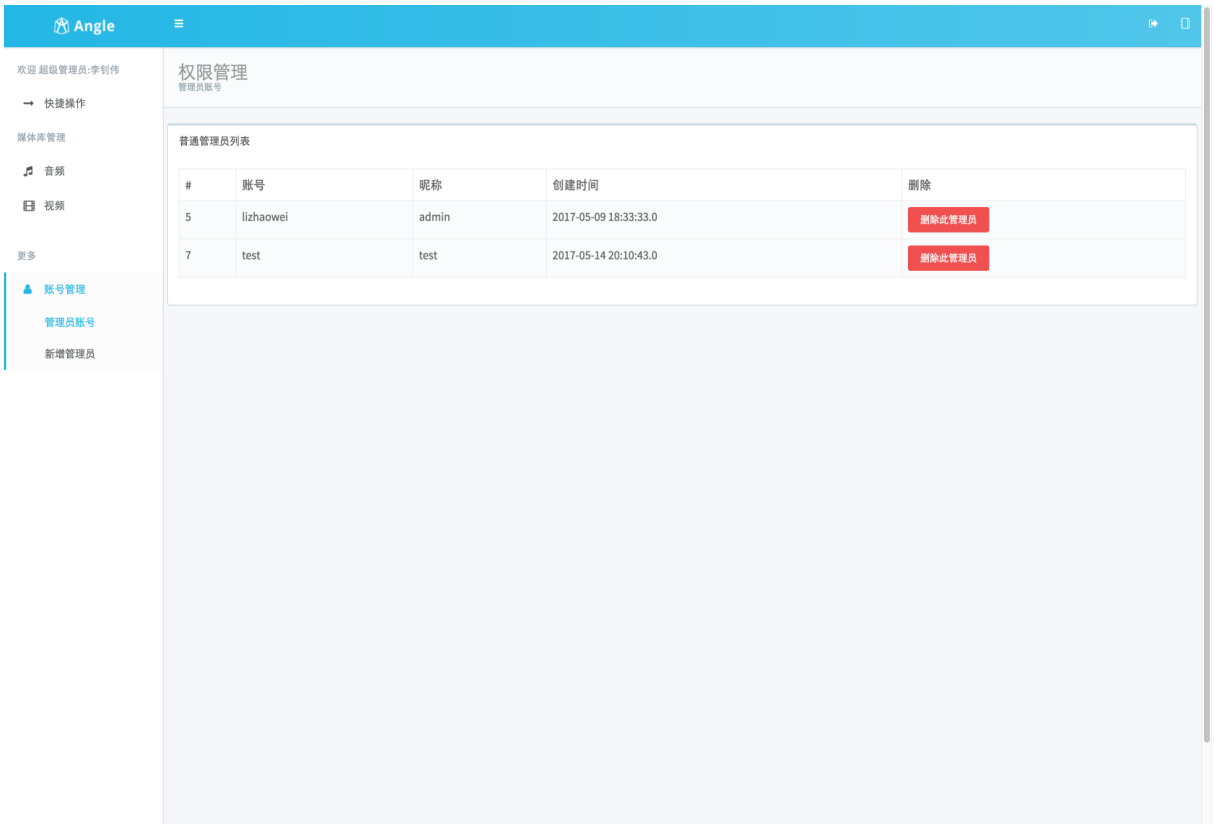


图 6-7 管理员管理页

（8）新增管理员账号（超级管理员身份下可见）

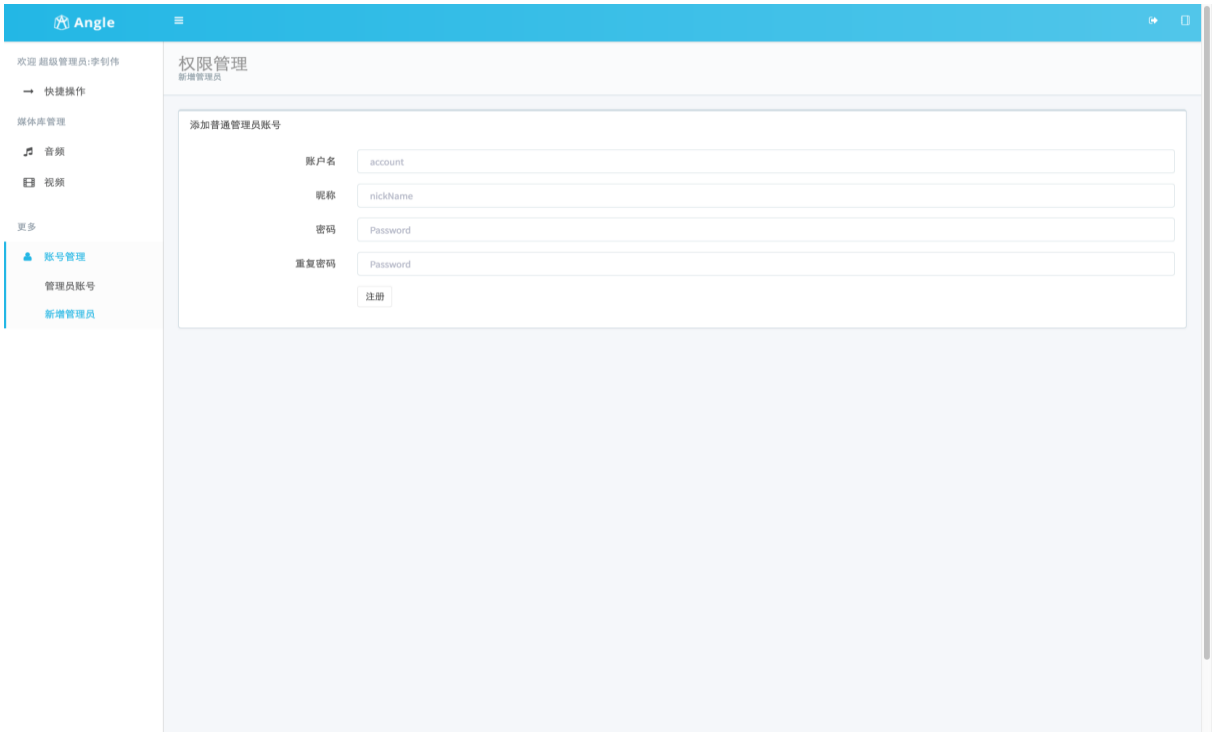


图 6-8 新增管理员页

（9）对于普通管理员，其不可见也无法访问“账号管理”页面

图 6-9 普通管理员无账号管理权限

6.1.2 客户端 Android 运行效果

（1）主侧滑菜单

图 6-10 主界面侧滑菜单

（2）本地音频列表

图 6-11 本地音频列表

（3）在线音频列表

图 6-12 在线音频列表

（4）音乐播放

进度条根据颜色的不同，区别播放进度、已缓冲进度。

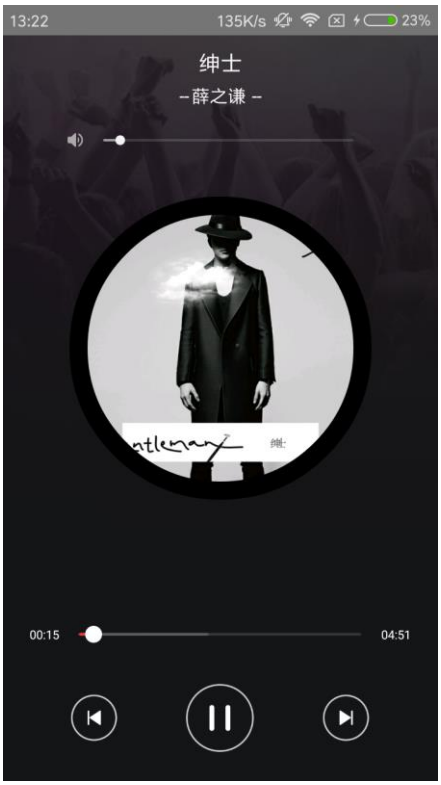


图 6-13 在线音频播放

（5）在线视频列表及列表中播放

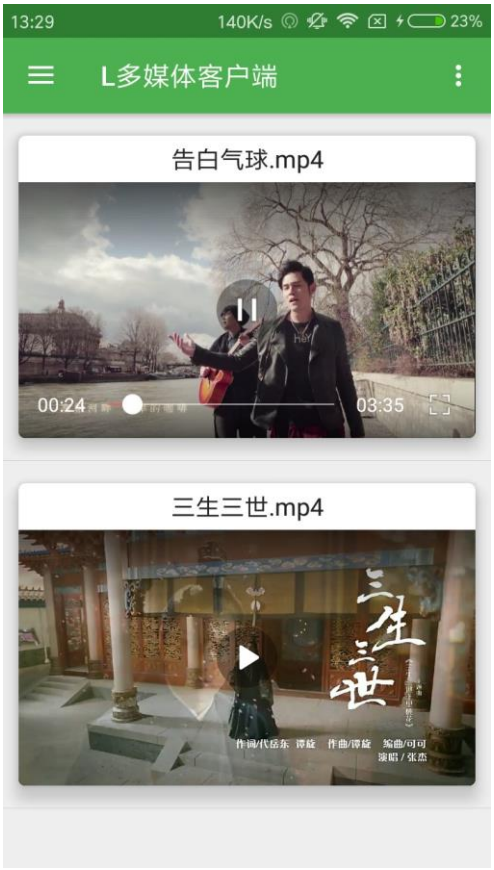


图 6-14 在线视频列表播放

（6）全屏播放

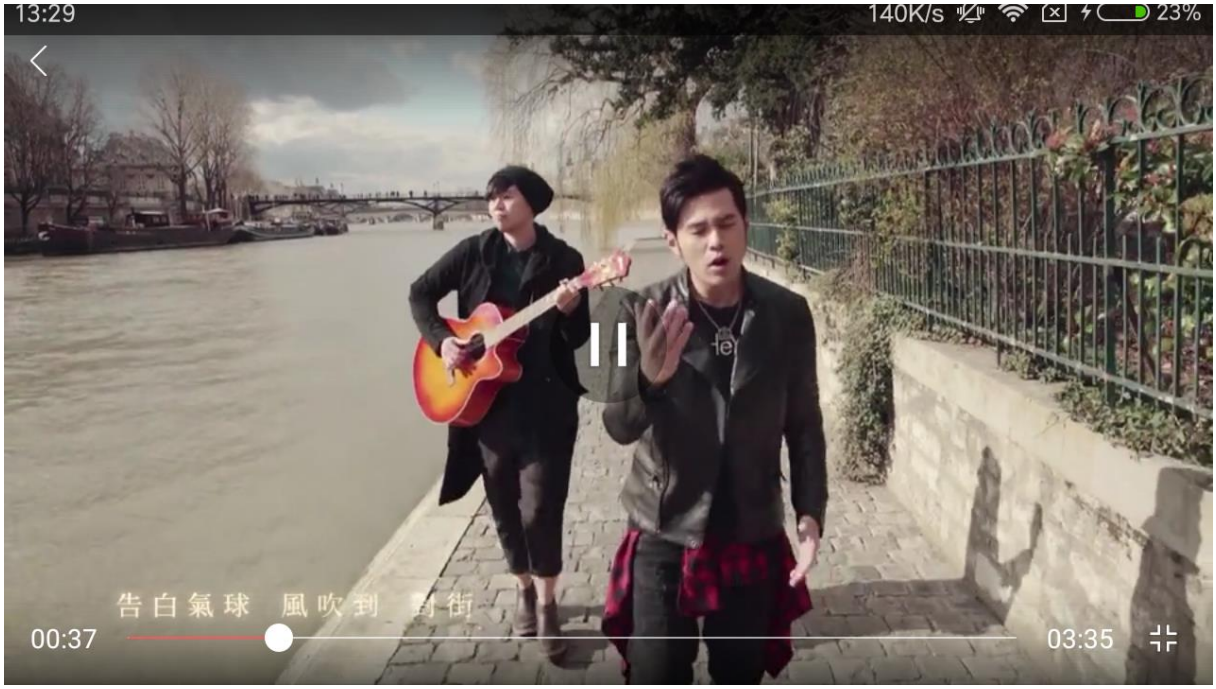


图 6-15 在线视频全屏播放

（7）管理员操作界面

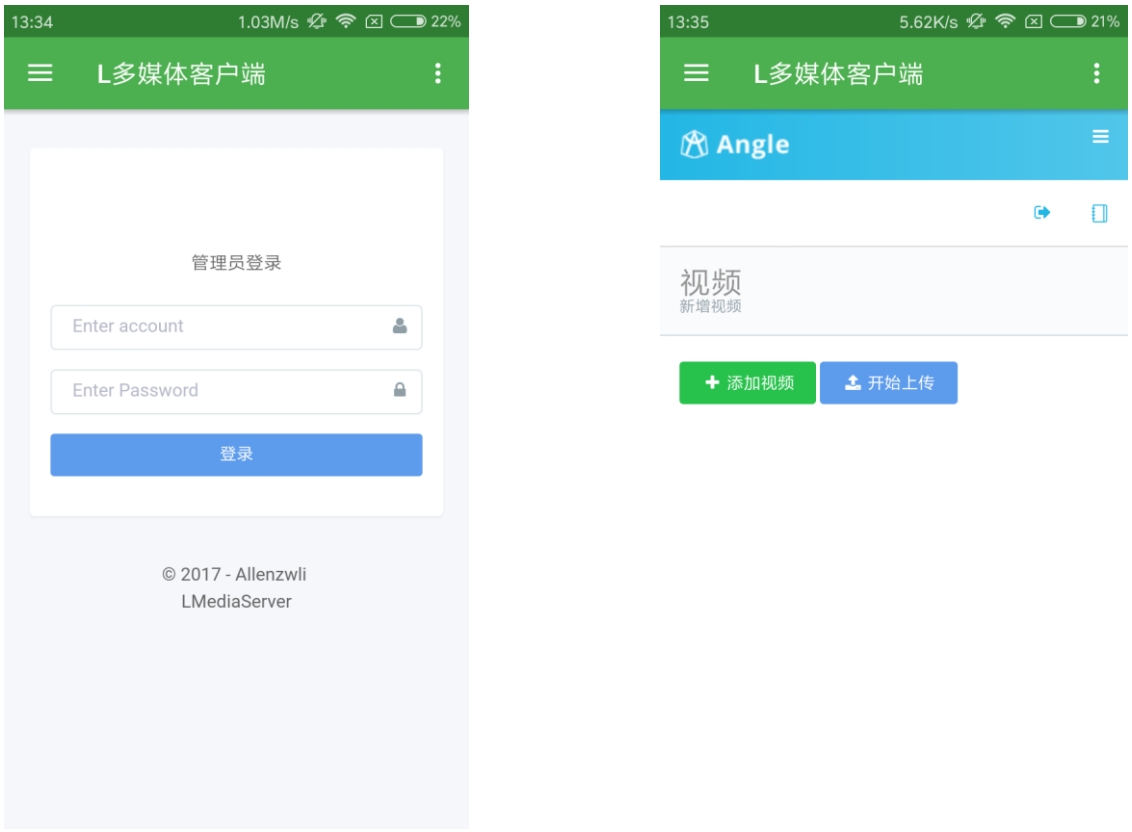


图 6-16 管理员操作

（8）上传文件选择（唤起系统文件管理器）

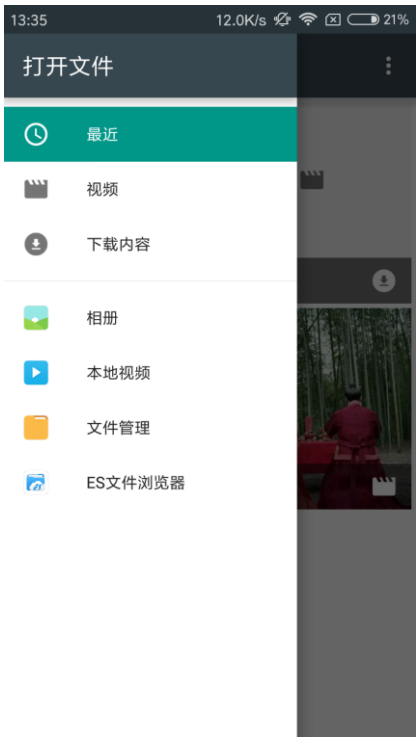


图 6-17 管理员文件选择

（9）文件下载（显示在通知栏）

图 6-18 文件下载

6.2 JUnit 单元测试

在 Web 应用开发中子模块开发完毕后需要进行测试时，可以选用 Junit 进行单元测试[17]。当系统的业务逻辑变得越来越复杂，开发人员在编写业务逻辑的具体代码中可能会无意遗留潜在的漏洞。为了能够及时的发现这类漏洞，测试人员可以选择 Junit 对某一个业务逻辑 Service 类中的业务方法进行单元测试，传入测试样例数据，并执行查看运行结果是否满足业务需求以及是否具有业务完备性。

在本 Web 应用的开发过程中，通过在 Maven 中添加 Junit 依赖，利用 InterlliJ IDEA 可以方便的建立单元测试。单元测试的代码都放在 test 路径下，并区别待测试的类所在的包名，如图所示。

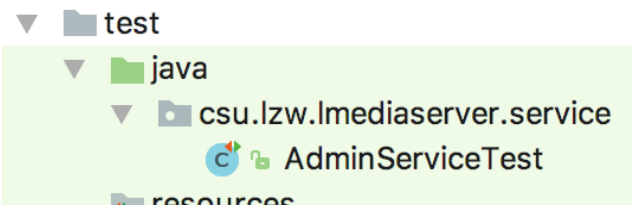


图 6-19 JUnit 单元测试目录

AdminServiceTest 为所创建的对于管理员业务逻辑的单元测试。

```
public class AdminServiceTest extends TestCase {
    public void setUp() throws Exception {
        super.setUp();
    }

    public void tearDown() throws Exception {
    }

    public void testGetAdmin() throws Exception {
    }

    public void testValidAdminToken() throws Exception {
    }

    public void testGetAllAdmins() throws Exception {
    }

    public void testDeleteAdmin() throws Exception {
    }

    public void testRegistAdmin() throws Exception {
    }

}
```

图 6-20 JUnit 单元测试

其中，setUp 方法负责对测试开始前所需要的对象进行初始化。tearDown 方法负责在测试结束后，将对象进行释放。以 test 开头的方法用来分别编写具体的业务逻辑测试代码。

6.3 小结

本章介绍了整个系统的运行结果以及单元测试方法。

第 7 章　性能优化

7.1 概述

在本项目中，不仅按照需求开发实现了各功能模块，而且在开发的过程中为了提高应用程序的性能，采取了一些方法。在以下分点详述。

7.2 Web 应用性能优化

（1）MySQL 建立索引

在通常的 Web 项目中，通常会有多条件查询或多调节排序的需求，并且可能是频繁的操作[18]。为了提高这一过程中的系统性能，可以通过在数据库中建立索引的方式，大大提高检索效率，减少耗时。MySQL 索引通常由两种数据结构实现，B+树和 Hash。在等值查询时，Hash 类型的索引查询速度更快；但如果是使用数值范围或 like 语句模糊查询时，应当使用 B+树类型的索引。

项目中的管理页面后台经常需要向数据库根据条件查找对应的文件列表，为了提高这一过程的查询速度，在 songs 表中 songName、aritst、album、fileName 字段建立索引；在 videos 表中 videoName 字段建立索引。提高多条件查询音频列表和视频列表过程中的系统性能。

（2）MyBatis 开启二级缓存

MyBatis 持久层框架本身提供二级缓存方法。

一级缓存作用于 SqlSession 级别，在 SqlSession 对象的内部有一个由 HashMap 构成的缓存区，如果有 SqlSession 一次执行了两条 SQL 语句，那么第一次执行完毕后，会把查询得到的数据写入 SqlSession 的缓存区域中，第二次的查询则不重新查询数据库，而直接从 SqlSession 对象的缓存区读取，降低了访问数据库的频率，提高了查询效率。本项目开发时整合 Spring 和 MyBatis 两个框架，SqlSession 对象的创建和释放是由 Spring 容器统一管理的。MyBatis 的一级缓存作用在同一个 SqlSession 对象内，并且默认开启[19]。

二级缓存作用于持久层的 Mapper 级别。如果有不同的多个 SqlSession 执行了同一个 Mapper 中的 SQL 语句，则将查询到的数据保存在二级缓存区域。二级缓存是跨 SqlSession 对象的，多个 SqlSession 对象可以共用使同一个 Mapper 下的二级缓存。二级缓存的开启需要在 Mybatis 的全局配置文件中配置，并在 Mapper 文件中通过

<cache></cache>开启。

Web 应用项目中，通过使用 MyBatis 二级缓存，减少了直接访问数据库的次数，提高了查询效率，优化了查询性能。

（3）前端页面使用 CDN 加速

CDN 即 Content Delivery Network 译为内容分发网络[20]·项目在前端页面的开发过程中，使用了 Bootstrap 和 jQuery 的 javascript 和 css 文件。而这些文件属于 Web 项目中的静态资源文件，如果只放在单个服务器上，受到单个服务器出站网络带宽的限制，一旦访问量变大，静态文件的出站速度会降低，获取静态文件的时间会变长，表现出来的就是网页加载变慢，卡顿等，用户体验很不友好。为了解决这个问题，通过在前端页面的 JS 和 CSS 声明时，使用 CDN 的 URI。根据 CDN 的机制，通用静态文件被缓存在众多分布式的服务器上，当用户访问页面，需要获取静态的 javascript、css 文件时，CDN 会根据用户的网络运营商或者用户位置，将距离用户最近并且能够提供最优网络传输服务的节点服务器上的静态资源传送给用户，从而有效的避免了网络拥挤，优化了前端页面的加载速度和系统响应时长。

7.3 Android 应用性能优化

（1）使用<merge>优化 View 绘制

在 Android 应用的开发过程中 View 的过度绘制是一个需要考虑并优化的问题。View 的过度绘制会在布局文件上的一个 View 创建时，在一帧的时间内其中的一个像素区域被重复地设置了很多次，使底层的机器 GPU 绘制了很多无用的图层，造成了无用的浪费，降低了系统性能。通过使用<merge>和<include>标签来在 XML 布局文件中定义 View 控件，可以减少布局嵌套，降低 View 所在层级。在 View 树遍历的时候，减少了不必要的 View 节点，减少了 GPU 无用的绘制，提高了应用程序的性能[21]。

在本项目中，加载进度圈使是一个通用的控件，在描述其布局时，单独建立 common_loading.xml 文件，在其他布局文件需要加载进度圈控件时，使用 merge 和 include 引入。

（2）在 ListView 中复用 Item

ListView 控件在使用时，其适配器中有一个 getView 方法，此方法需要将列表中每一项 ItemView 作为返回值返回。如果不去复用这些 ItemView,那么每次在列表滑动过程中，当有新的 Item 滑出时，都需要重新创建 ItemView，而创建的过程是一个耗资源的，

耗时的过程，表现出来的就是，当列表中的元素个数如果成千上万非常多时，滑动的操作变得不流畅，变得缓慢。为了解决这个问题，在开发 ListView 的适配器时，建立 ViewHoler，复用 ItemView，减少新创建 ItemView，提高应用程序的滑动性能。

具体做法如下：

```
@Override
public View getView(int i, View view, ViewGroup viewGroup) {
    MusicViewHolder holder=null;
    if(view==null){
        view= LayoutInflater.from(mContext).inflate(R.layout.item_music, null, false);
        holder=new MusicViewHolder(view);
        view.setTag(holder);
    }else{
        holder= (MusicViewHolder) view.getTag();
    }
    final Song song = mSongBeanLists.get(i);
    holder.songNameTV.setText(song.getSongName());
    holder.artistTV.setText(song.getArtist());
    holder.albumTV.setText(song.getAlbum());
    Glide.with(mContext).load(song.getPictureUrl())
            .error(ATEUtil.getDefaultAlbumDrawable(mContext))
            .placeholder(ATEUtil.getDefaultAlbumDrawable(mContext))
            .diskCacheStrategy(DiskCacheStrategy.SOURCE)
            .centerCrop()
            .into(holder.albumIV);
    return view;
}

class MusicViewHolder {
    TextView songNameTV;
    ImageView albumIV;
    TextView artistTV;
    TextView albumTV;

    public MusicViewHolder(View contentView) {
        songNameTV= (TextView) contentView.findViewById(R.id.songNameTV);
        artistTV= (TextView) contentView.findViewById(R.id.artistTV);
        albumTV= (TextView) contentView.findViewById(R.id.albumTV);
        albumIV= (ImageView) contentView.findViewById(R.id.albumIV);
    }
}
```

图 7-1 ListView 复用 Item 方法

7.4 可扩展的性能优化

（1）概述

除了上述已经应用于项目中的性能优化方法外，还有以下方法可以进一步优化应用性能。鉴于本项目仅为毕业设计，并且需求较为单一，以及个人开发应用的成本限制，我并没有将下列性能优化的方法用于此项目中去。但对于 Web 应用开发，以下方法也是应当考虑使用的。

（2）Redis 缓存

Redis 是一个键值型的内存数据库[22]，具有性能高、持久化、主从复制块、自动操作等特点。在高并发的系统中，把需要经常查询的数据存储在 Redis 内存数据库中，极大地提高了数据访问的效率，减轻了数据库的访问压力，提高了系统的处理能力。

（3）Nginx 与 Tomcat 集群以及负载均衡

服务器集群可以提供比单一服务器具有更高可扩展性和可用性的服务平台，并且具有负载均衡和错误恢复的能力。在 Java Web 应用中通常通过 Nginx 和 Tomcat 结合使用实现，通过配置 Nginx，将请求任务分配到多台机器上，减轻单一服务器的访问压力，提高了系统的处理能力和可重用性，并具有错误恢复的能力。

7.5 小结

本章介绍了在开发过程中使用的性能优化方法，以及提出了可以继续扩展的性能优化方法。

第 8 章　总结与展望

8.1 总结

本文详述了毕业设计多媒体在线管理系统的开发过程，包括需求分析、概要设计、详细设计、运行与测试、性能优化等内容。重点叙述了项目各模块的开发设计方案和技术要点以及应用程序性能优化的方法。本次毕业设计包含服务端和客户端，开发内容涵盖了数据库设计、Web 后台开发、Web 前端开发、Android 客户端开发，具体包含前端 jsp 页面设计与编写、CSS 样式编写、javascript 脚本编写、服务端后台 Java 代码编写、数据库设计与 SQL 语句编写、客户端 Android 界面设计与 XML 布局编写、Android 应用开发的 Java 代码编写，所有工作由本人独立全栈式开发完成。通过完成此次毕业设计，温习并整合了使用 Java 语言开发 Web 应用和 Android 应用程序的开发技术。为了构建低耦合的应用程序，提高可扩展性和可复用性，项目不仅使用了 MVC、MVP 等设计理念进行架构，而且多处使用了设计模式，如策略模式解析音频文件信息、观察者模式处理手机网络变化情况，单例模式实现音频播放列表等。为了构建高性能的应用程序，项目整合并使用了 Java Web 开发的三大框架 Spring MVC 框架、Spring 框架、Mybaits 框架，通过数据库连接池、缓存、Spring 容器管理等方法，提高了系统的性能。本着开源分享的精神，本人已将项目源代码开源到 Github 上，希望在完成本毕业设计的同时能够为后来者在开发类似多媒体管理系统时提供解决方案参考。

此外，通过完成此次毕业设计项目，我体会到毕业设计是一个将学校里四年所学的专业知识与实际的应用相结合的锻炼机会。当下正是互联网快速发展的时代，作为一名通信工程专业的学生，能够结合时代需求，具有一定程度的互联网应用开发专业技术是必不可少的。在完成本次毕业设计项目的时候，遇到问题，我能够独立思考并得出解决方案，并通过程序代码将具体的解决方案实现，以实现毕业设计项目业务需求。对于一名即将毕业成为工程师的应届生而言，我认为这是一种基本的、必要的工程能力。在以后的开发工作中，我将会不断提高自己的技术深度，提高解决问题的能力。

8.2 展望

对于本项目而言，还可以再优化设计的地方是 Android 端管理员操作模块，目前是通过在 Fragment 中使用 WebView 控件的方式实现，其本质是在 Android 端操作服务端提供的 Web 网页，如果当网络状况不好或服务器访问压力大时，在 Android 客户端管理

员对文件进行管理操作的过程中会出现卡顿不流畅的现象。可以进一步改进为使用 Android 原生的控件进行界面实现，通过 Android 客户端向服务端后台接口发送 HTTP 请求来实现，从而提供更好的用户体验。此外，本项目前端页面开发使用的技术是流行已久的 jQuery 和 Bootstrap。对于当下前端开发的热点，现代化框架层出不穷，如 AugularJs、Vue.js、React.js 等，其中用到的面向组件编程以及 MVVM 数据绑定模式等具有更优雅的设计。在未来的项目中，应当紧跟时代发展潮流，不断学习新技术、扩展自己的技术栈从而更好地提高应用开发的能力。此外，尽管 Java Web 开发的三大框架 Spring MVC 框架、Spring 框架、MyBaits 框架尽管拥有极好的性能和可扩展性，但对于开发人员，不能仅仅停留在使用工具的层面，对于框架内部的原理和架构，应当进一步学习与挖掘，如 Spring 的依赖注入(DI)、控制反转(IOC)、面向切面编程(AOP)等优秀的架构设计方法，在此基础上，开发人员应当在理解并熟悉后，自己尝试独立构建一个可扩展的、高性能的、可以提供给其他开发者使用的类库或框架，提高自己的软件架构能力和影响力。

致谢

时光如白驹过隙一般，转眼间，大学即将毕业。在毕业论文完成之际，我首先要感谢我的导师彭春华老师对于本论文在写作时的细心指导，以及老师四年来对我的帮助和支持。此外，在即将毕业离校之际，我要感谢中南大学信息科学与工程学院四年来对我的教导，我会铭记每一位任课老师的教诲，熟练掌握老师们所传授的专业知识，并将其用于以后的实践应用中去。感谢中南大学通信工程 1301 班的每一位同学，感谢这四年里每一位同学对我的帮助，这份友谊我将铭记在心。最后，我要感谢我的父母，感谢他们对我的理解与支持，让我能坚持自己的想法，选择自己毕业后的人生道路。我的学生时代即将结束，将要进入企业，成为一名开发工程师，参与大型项目，开发商业产品。对此，我将会继续提高自己的技术深度，扩展自己的技术栈，增强自己的工程架构能力。我将会怀着理想信念，一步一步在今后继续努力，让自己变得更加优秀，实现自己的人生价值，为祖国为社会做出自己应有的贡献。

希望我能成为我最想成为的人，希望前面这句话不只是希望。

参考文献

[1] 艾默拉. 你不知道的 5 大 Linux 发行版[J]. 计算机应用文摘, 2011(12):18-19.

[2] 张洪伟. Tomcat Web 开发及整合应用[M].北京: 清华大学出版社, 2006.

[3] 李立功, 赵扬. MySQL 程序设计与数据库管理[M].北京: 科学出版社, 2001.

[4] 罗文龙. Android 应用程序开发教程:Android Studio 版[M]. 北京:电子工业出版社, 2016.

[5] 任海鹏, 邓春红, 汪学文. Android 操作系统兼容性测试系统研究[J]. 绥化学院学报, 2016(3):141-143.

[6] 吴安. 基于 MVC 设计模式的系统框架研究与设计[D]. 江苏大学, 2009.

[7] 舒礼莲. 基于 Spring MVC 的 Web 应用开发[J]. 计算机与现代化, 2013(11):167-168.

[8] 胡启敏, 薛锦云, 钟林辉. 基于 Spring 框架的轻量级 J2EE 架构与应用[J]. 计算机工程与应用, 2008, 44(5):115-118.

[9] 荣艳冬. 关于 Mybatis 持久层框架的应用研究[J]. 信息安全与技术, 2015, 6(12):86-88.

[10] 张锦贤. 基于 bootstrap 框架的 web 设计开发研究[J]. 工程技术:文摘版, 2016(8):00089-00089.

[11] 何国英, 高炜. 基于 AJAX 和 CSS 技术的教师在线评价系统设计[J]. 昆明学院学报, 2013, 35(6):109-111.

[12] 张娜. Android 系统架构研究与应用[D]. 西安科技大学, 2013.

[13] 董晓刚. 浅析 Android 系统的四大基本组件[J]. 中国电子商务, 2013(1):39-39.

[14] 杨靖, 洪蕾. MVP 设计模式研究及在 Android 中的设计[J]. 科技创新导报, 2016, 13(34):97-98.

[15] 孟远. Android 网络通信框架 Volley 的解析和比较[J]. 软件, 2014(12):66-68.

[16] 李爽. Session 安全性研究及应用[J]. 电子世界, 2013(23):16-16.

[17] 方梁. 创建自动运行的 JUnit 单元测试框架[J]. 程序员, 2006(7):98-102.

[18] 王正万. MySQL 索引分析及优化[J]. 凯里学院学报, 2006, 24(3):54-55.

[19] 董欣欣. 基于 SSM 的大讲堂后台管理系统的设计与实现[D]. 大连理工大学, 2016.

[20] Peng G. CDN: Content Distribution Network[J]. Research Proficiency Exam Report, 2004:1-6.

[21] 张延年, 米洪. Android 应用开发中 ListView 组件性能优化的研究[J]. 现代计算机, 2015(24):58-64.

[22] Kreibich J A, Sanfilippo S, Noordhuis P. Redis: The Definitive Guide: Data Modeling, Caching, and Messaging[J]. Oreilly Media Inc Usa, 2011(3):54-55.