华东交通大学

毕业设计（论文）

题目：基于 SSM 的抒情电影网站的设计与实现

学　　院:	软件学院		
专　　业:	软件工程	班　　级:	软件工程(传习)2016-2
学生姓名:	刘桂华	学　　号:	2016211001001419
指导教师:	李文奇	完成日期:	2020 年 4 月 30 日

毕业设计（论文）诚信声明

本人郑重声明：所呈交的毕业设计（论文）是我个人在导师指导下进行的研究工作及取得的研究成果。就我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表和撰写的研究成果，也不包含为获得华东交通大学或其他教育机构的学位或证书所使用过的材料。

如在文中涉及抄袭或剽窃行为，本人愿承担由此而造成的一切后果及责任。

本人签名

导师签名

2019 年 **4** 月 **30** 日

华东交通大学毕业设计（论文）任务书

姓名	刘桂华	学号	2016211001001419	毕业届别	2020	专业	软件工程
毕业设计（论文）题目	基于 SSM 的抒情电影网站的设计与实现						
指导教师	李文奇	学历	本科	职称	工程师		

具体要求：

1、设计目的与内容

设计目的：提供一个平台给大家抒发对电影的看法，并能够给这些电影赋予相应的星级评分。让每位观众都能在以线上的方式抒发自己对电影的想法和意见，从而更好的与其它观众互动。

设计内容：设计并实行一个具有电影展示、电视剧吧展示、电影分类、电视剧分类、电影搜索、电视剧搜索、电影评分排行榜、电影评论、收藏电影、电视剧评论、收藏电视剧、收藏列表、已评论列表、个人信息管理、后台管理等的网站。

2、设计要求与成果

设计工具：IDEA

技术要求：SSM 框架实现后端部分，jsp,js,jQuery，ajax 实现前端页面，MySQL 数据库做数据存储。

设计步骤：确定需求，划分模块，设计页面，后台逻辑编写，测试

设计成果：毕业论文，源码

3、应收集的主要参考文献

[1] 杭建. Java 工程师修炼之道[M]. 北京：电子工业出版社，2018.

[2] Caliskan . Sevindik.Beginning Spring[M].北京:清华大学出版社,2015.

[3] 张甦. MySQL 王者晋级之路[M]. 北京：电子工业出版社，2018.

[4] 姜桂洪. MySQL 数据库应用与开发[M]. 北京：清华大学出版社，2018.

[5] 阳波. JavaScript 核心技术开发解密[M]. 北京：电子工业出版社，2018.

[6] 刘增辉. MyBatis 从入门到精通[M]. 北京：电子工业出版社，2017.

[7] 姜桂洪. MySQL 数据库应用与开发[M]. 北京：清华大学出版社，2018.

进度安排：

	设计各阶段内容	时间	%
1	选题和资料收集阶段	第 7 学期 16～18 周	5
2	分析计划阶段	第 7 学期 19～20 周	10
3	设计阶段	第 8 学期 1～3 周	25
4	实现和测试阶段	第 8 学期 4～7 周	40
5	毕业论文写作、查重、答辩	第 8 学期 8～13 周	20
6	英文资料翻译	第 7 学期 19～20 周	

指导教师签字：

2019 年 12 月 15 日

题目发出日期	2019.12.16	设计（论文）起止时间	2019.12.16-2019.6.7

学院意见：

同意发布题目

毕业设计领导小组组长签章

华东交通大学毕业设计（论文）开题报告书

课题名称	基于 SSM 的抒情电影网站的设计与实现				
课题来源	自拟	课题类型	AY	导师	李文奇
学生姓名	刘桂华	学号	2016211001001419	专业	软件工程

一、开题报告内容

1、目的和意义

目的：随着科技的进步，人们的生活水平逐渐提高，大家的生活越变的丰富多彩，就拿电影来说，人们偶尔吃完晚饭，就可能会和家人一起观看电影，当然，一些情侣出来玩，电影也是必不缺少的一部分，既然观看完了电影，那么人们就有相应的看法需要讨论，那么就应该有一个平台给大家抒发对电影的看法。

意义：为电影提供相应的评价，能够促进大家之间的沟通，对于一些比较害羞的人，他们平时不敢在实际生活中表达自己的已经，那么他就可以在这个平台上抒发自己的意见。

2、研究现状和发展趋势

研究现状：随着网络水平的飞速发展，人们对网络的使用也变得越来越熟悉，只要社会的进步，人们就会对生活有所要求，观看电影当然是必不可缺少的，人们相应的就会产生相应的看法和想法。同样的，国内外也有许多影评网站，如：ROTTEN TOMATOES、IMDb、好莱坞、豆瓣影评、迅雷看看、大众影评网、时光机等。

发展趋势：目前国内有关电影的网站确实有许多，但是出名的网站也没多少，目前了解到的影评网站有豆瓣影评、迅雷看看、大众影评网、时光机等。但是各有各自的缺点和优点，不过随着社会的进步影评网站能够做的越来越好。

3、设计方案

开发语言：Java

后台开发框架：SSM

前端页面：jsp，js,jQuery，Bootstrap，ajax

数据库：MySQL

开发模式：MVC

本平台包含用户和管理员两类角色。需要实现如下主要功能：电影展示、电视剧吧展示、电影分类、电视剧分类、电影搜索、电视剧搜索、电影评分排行榜、电影评论、收藏电影、电视剧评论、收藏电视剧、收藏列表、已评论列表、个人信息管理、后台管理等。

课题类型：（1）A—工程设计；B—技术开发；C—软件工程；D—理论研究；

（2）X—真实课题；Y—模拟课题；Z—虚拟课题

（1）、（2）均要填，如 AY、BX 等。此部分可以附页

华东交通大学毕业设计（论文）开题报告书（续）

4、进度安排

	设计各阶段内容	时间	%
1	选题和资料收集阶段	第 7 学期 16～18 周	5
2	分析计划阶段	第 7 学期 19～20 周	10
3	设计阶段	第 8 学期 1～3 周	25
4	实现和测试阶段	第 8 学期 4～7 周	40
5	毕业论文写作、查重、答辩	第 8 学期 8～13 周	20
6	英文资料翻译	第 7 学期 19～20 周	

5、主要参考文献

[1] 肖睿,吴振宇. BOOTSTRAP 与 JQUERY UI 框架设计[M].北京:中国水利水电出版社,2017

[2] 贺春旸. MYSQL 管理之道:性能调优、高可用与监控[M]. 北京：机械工业出版社，2016.

[3] 张甦. MySQL 王者晋级之路[M]. 北京：电子工业出版社，2018.

[4] 姜桂洪. MySQL 数据库应用与开发[M]. 北京：清华大学出版社，2018.

[5] 阳波. JavaScript 核心技术开发解密[M]. 北京：电子工业出版社，2018.

[6] 刘增辉. MyBatis 从入门到精通[M]. 北京：电子工业出版社，2017.

[7] 姜桂洪. MySQL 数据库应用与开发[M]. 北京：清华大学出版社，2018.

[8] 张甦. MySQL 王者晋级之路[M]. 北京：电子工业出版社，2018.

二、方法及预期目的

1、研究方法（手段）

基于现有的电影评论平台，通过调研了解用户需求写出需求分析、划分软件模块。查阅 SSM，MySQL 等相关资料，通过使用 java 的 web 框架 SSM 与 MySQL 数据库相结合完成设计，并对系统进行测试。

2、预期目的

平台功能全面、操作简单，用户可以在线评论电影，管理员能够发布电影。具体目标如下：电影展示、电视剧吧展示、电影分类、电视剧分类、电影搜索、电视剧搜索、电影评分排行榜、电影评论、收藏电影、电视剧评论、收藏电视剧、收藏列表、已评论列表、个人信息管理、后台管理等功能。

三、指导老师意见

同意开题

指导教师签名：

日期：2019.1.10

基于 SSM 的抒情电影网站的设计与实现

摘要

在已跨入 21 世纪的今天,人类使用和学习信息的方式以及信息的包装方式正在进行着不可阻挡的革命。目前,我国上网的人口已经过亿,是世界上网民最多的国家,许多人在需要查询信息,首先想到的就是上网。网站的迷人之处在于综合使用文本、图象、声音、动画和电影院的信息和内容,具有丰富的多媒体表现与互动特点,无可置疑,网站已成为最吸引人的也最有效的信息传递手段和方式。随着计算机技术和 Internet 的日新月异,电影评论网站技术因良好的人机交互性倍受教育、娱乐等行业青睐。

本网站充分运用网页编程的一些基本特点,加入了 Java 语言、Javascript 代码与 CSS 脚本等页面元素。该网站前端页面主要采用 JSP 页面，后台数据库是使用 MySQL，前台通过采用 AJAX 访问后端接口，从而达到前后端的联调工作，对于一些数据多的页面采用分页处理等。本网站图文并茂、界面直观、操作简单,内容布局条理清楚,颜色鲜明、搭配合理,内容丰富,令您在测览本站时能得知最新的电影资讯,还可以发表对电影的评论。

本电影评论网站系统在 WEB 的基础上模拟企业级的电影评论网站系统,实现用户信息管理、电影信息的添加、删除、修改、电影的星级评分评论、搜索电影、电影高分排行榜等功能。由于本系统是一个小型系统,所以本人采用基本的 MySQL 数据库,易于实现。具体实现中将 HIML、JSP 及 Javascript 完美融合,力求界面美观、操作流畅。

关键词：Spring；MVC；MySQL；Ajax

Design and implementation of lyric movie website based on SSM

Abstract

In the 21st century, the way of using and learning information and the way of packaging information are undergoing an unstoppable revolution. At present, the population of Internet users in China has exceeded 100 million, and it is the country with the largest number of Internet users in the world. Many people need to query information, and the first thing they think about is Internet access. The charm of the website lies in the comprehensive use of text, image, voice, animation and Cinema Information and content, with rich multimedia performance and interaction characteristics. Undoubtedly, the website has become the most attractive and effective means and means of information transmission. With the rapid development of computer technology and Internet, the technology of movie review website is favored by education, entertainment and other industries because of its good human-computer interaction.

This website makes full use of some basic features of web programming, and adds page elements such as Java language, JavaScript code and CSS script. The front-end page of the website mainly uses JSP page, the background database uses mysql, the front-end uses Ajax to access the back-end interface, so as to achieve the front-end and back-end joint debugging work, and page processing is used for some pages with more data. This website is featured with both pictures and texts, intuitive interface, simple operation, clear content layout, bright color, reasonable collocation, rich content, so that you can know the latest movie information when you are browsing this website, and you can also comment on the movie.

On the basis of web, this movie review website system simulates the enterprise level movie review website system, and realizes the functions of user information management, movie information addition, deletion, modification, star rating review of movies, search for movies, movie high score ranking, etc. Because this system is a small system, so I use the basic MySQL database, easy to achieve. The specific implementation will integrate himl, JSP and JavaScript perfectly, and strive for beautiful interface and smooth operation.

Keywords: Spring；MVC；MySQL；Ajax

目录

1 绪论

1.1 研究的背景及意义

1.1.1 选题的背景

如今的社会，科技发展速度越来越快，大家的生活水平飞速的提高，对互联网的应用也十分依赖，同时产生一系列的娱乐活动，如轰趴、野外露营、KTV、看电影等。科技的飞速发展，让人们的越来越会懂得如何享受生活。那么，看电影也就成为必不可少的一个娱乐活动。大家吃完晚饭和家人出去看看电影，无聊之时也可和朋友一起去看看电影，对于一些新上映的电影，大家更喜欢去观看，当然，还一个更大众化的现象，当一对情侣出来玩耍之时，观看电影就成必不缺少的一项活动。

所以说，观看电影是十分流行的。而且会随着科技的飞速发展，看电影的人数就会越来越多哦！当然看电影的人数越多，相应的大家对电影也会产生不同的意见，那么，应用好互联网，就可在家对该电影进行评论，从而就出现影评网站。

1.1.2 国内外研究现状

随着科技的进步，人们的生活水平逐渐提高，大家的生活越变的丰富多彩，就拿电影来说，人们偶尔吃完晚饭，就可能会和家人一起观看电影，当然，一些情侣出来玩，电影也是必不缺少的一部分，既然观看完电影，那么人们就有相应的看法需要讨论，人们就可以在口头或者网上就行评论！那么现在国内也出现一系列的影评网站，如：豆瓣影评、时光网、大众影评网、猫眼等。同样的，这些网站也是大部分人寻找高分电影的好处，大家同也喜欢在这些网站对自己看过的电影进行评论。

当然，在国外也有着比较好的影评网站，如：IMDb、Metacritic、Rotte Tomatoes 等，对于这些网站如果热爱电影应该都会知道。对于许多人都宁愿去国外网站而不愿意使用国内网站感到心酸。

但是，就现状来说，国内影评网站在逐渐退步，这是一个不好的现象。

随着科技的飞速发展，互联网上既出现好的东西，同样也产生一些坏的东西。就拿电影评论网站来讲，许多无所事事的键盘侠们无缘无故的瞎评论，从而导致电影评论失去真实性，同样，也有一群人会花钱找一群水军，在该电影下疯狂给好评，这样做也失去对电影评论的公平性。

1.1.3 研究的意义

想让国内的电影评论得到飞速的发展，想让这类网站在世界也占有一席地位。想让电影评论网站拥有它的真实性，拥有它的公平性，而不是一个谋取利益的地方。不想让这种事情进行再一步的恶化，想让国家发展得更好。

1.2 系统目标

做这个电影评论网站的目标就是想为让大家能够更好的表达自己对一部电影的看法，想通过互联网就可以时时刻刻的观察一部电影的最新状况。还有就是希望少一些水军参与电影评论，让它拥有它的公平性和真实性，从而能更好地为大家提供最重要的帮助。

2 需求分析

2.1 功能需求

通过在网上的查找资料，合理的划分为这几个模块：电影展示、电视剧展示、电影分类、电视剧分类、电影详情、电视剧详情、收藏列表、已评论列表、个人信息管理、后台管理等功能。电影展示和电视剧展示主要是在首页展示一系统的电影和电视剧，并以轮播图的样式展现出来，当然，也可以查看所有的电影和电视剧，同样，也可以在搜索栏通过模糊查询进行搜索电影和电视剧，首页那里还有一个高分电影排行榜，按照电影评分的高低进行电影前十排行，在所有电影和电视剧里面，也将电影进行分类处理，更好增加用户的体验；电影分类和电视剧分类主要实现是在上面的导航栏有一个分类查询，大致的将电影和电视剧分为几类，用户可以通过点击电影类别查看所有该类的电影，简洁而快速；电影详情和电视剧详情主要是点击一个电影里面去，用户就可以查看到有关的详细信息展示，同样，用户也能观看该电影或电视剧的所有评论，而且用户需要可以在该页面进行评论等功能，同样的，该模块还有添加收藏功能，收藏电影和收藏电视剧是在电影详情页增加一个添加最喜欢的按钮，通过点击这个按钮可以将该电影或电视剧加入我最喜欢的收藏列表里面去；已评论列表是在用户管理里面可以查看到自己对电影和电视剧的评论；收藏列表则是可以在用户管理里面可以查看到自己添加的所有最喜欢的收藏，在该列表里面也可以通过点击进入电影或电视剧详情页；个人信息管理就是用户在个人信息里面观看自己的信息，同时也可以修改自己的信息；后台管理模块是管理员的操作，可对电影进行增加、删除、修改。

2.1.1 子系统/模块说明

前台：用户登录、注册、修改个人信息、修改密码。查看电影列表、查看电视剧列表、查看电影详情、查看电视剧详情、搜索电影、搜索电视剧、收藏电影、收藏电视剧、评论电影、评论电视剧、查看收藏列表、查看评论电影和电视剧、查看电影排行榜。

后台：修改管理员密码、添加电影、删除电影评论、删除电视剧评论、删除用户。

用户在使用该网站使用是不同的权限。主要分为三种权限：未登录的游客、登录后的用户、管理员，他们之间的权限将以下列事以图展示出来。

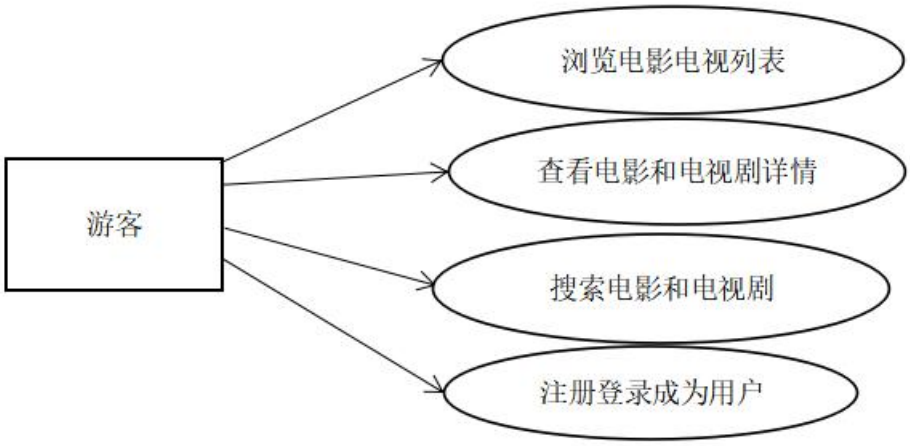

图 2-1 游客用例图

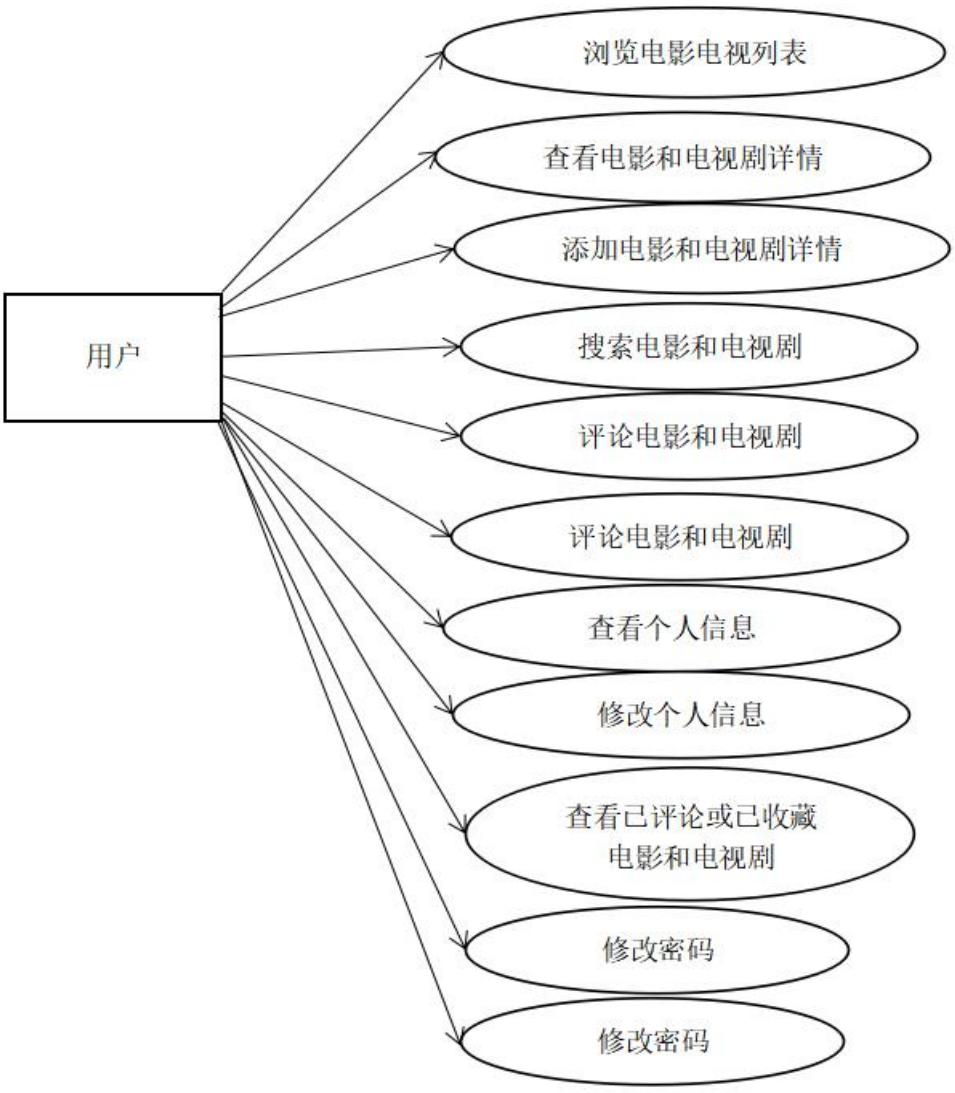

图 2-2 用户用例图

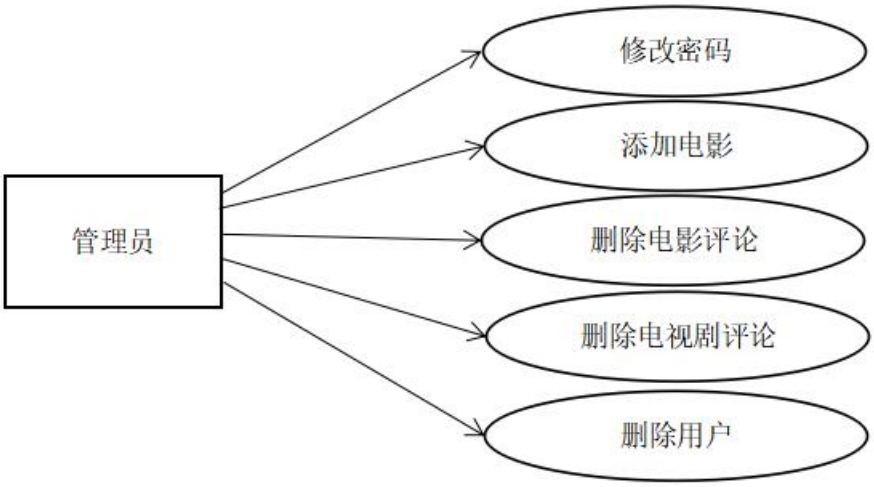

图 2-2 管理员用例图

2.1.2 功能需求描述

（1）电影展示和电视剧展示

电影展示和电视剧展示模块主要包含电影展示电影展示、电视剧展示、评分前十排行榜、搜索电影和搜索电视剧。

电影展示和电视剧展示都分为几类：首页以轮播图的展示、全部电影和全部电视剧以列、半平铺、全平铺的方式展示。

评分前十排行榜是通过提供的对这些电影的评分进行一个排名，并将前面十名展示出来。

搜索电影和搜索电视剧是通过搜索栏搜索自己想要的电影，该功能是通过模糊查询实现。

（2）电影分类和电视剧分类

电影分类和电视剧分类是将这些电影电视剧分为大致的以下七类：剧情、动画、爱情、奇幻、犯罪、惊悚、动作。用户通过挑选自己喜欢的类型，即可找到相应的电影和电视剧。

（3）电影详情和电视剧详情

电影详情和电视剧详情主要包含以下信息：电影或电视剧封面，电影或电视剧名称，电影或电视剧总评分、电影或电视剧概述、电影或电视剧主演名单、电影或电视剧评论总条数，该电影或电视剧的所有评论、对电影或电视剧进行添加到最喜爱的里面去。

（4）电影评论和电视剧评论

电影评论和电视剧评论是在电影详情页进入的，进行评论需要先进行登录。评论主要分为三个部分：标题、星级评分、内容。当评论提交之后就会在该页面

回显出来，回显出来的内容包含标题、星级评分、内容、评论时间、评论作者等。

（5）个人信息管理

个人信息管理是需要在登录之后才可通过点击用户名进去，该模块主要包含：个人信息、修改个人信息、修改密码、查看自己所评论的电影和电视剧、查看自己所收藏的电影和电视剧。个人信息主要包含用户名、邮箱、名字、姓、家庭地址；然而修改信息只能修改名字、姓、家庭地址；修改密码则需要输入两次密码进行确认，确认通过即可；查看自己所评论的电影和电视剧可以看到自己所评论过的所有电影和都电视剧，同时可以看到自己所评论的内容等；查看自己所收藏的电影和电视剧可以看到自己所收藏过的所有电影和都电视剧，也可点击电影或电视剧进入相应的电影或电视剧详情页。

（6）后台管理

后台管理模块主要包含修改管理员密码、添加电影、删除电影评论、删除电视剧评论、删除用户。修改管理员密码首先需要确认两次密码，确认成功即可修改成功；删除电影评论则是管理员删除一些不怎么符合规定的电影评论；删除电视剧评论则是管理员删除一些不怎么符合规定的电视剧评论；删除用户则是管理员删除一些已弃用或者违反规定的账号；添加电影则是管理员根据电影新片添加一些新上映的电影，主要需填写电影名字、电影封面、电影类别、电影简介、电影评分字段，最后点击提交即可添加进去。

2.2 非功能需求

2.2.1 外部插件需求

由于对于电影评论和电视剧评论需要用到星级评分，特意在外面找来星级评分的插件，最后集成插件，并且进行调用。将该插件的 js 和 css 包进行导入，再通过调用 js 方法进行调用它，即可使用该插件，若要做修改，即可在 js 和 css 里面修改。

对于展示全部电影或电视剧的时候等，需要对相应的记录做分页处理。于是乎，就找到相应的分页插件 pagehelper。将该插件配置好，集成到系统里面，使得开发更方便与快捷。

2.2.2 性能需求

完整性约束：对于一些信息的填写，当一些数据字段不能为空的时候，用户操作的时候应该会提出提示。

安全性约束：当一些功能在用户未登录之时，游客是不允许进行相应的操作的。例如：当未登录的时候，用户进不去查看个人信息模块、用户对电影或电视剧实现不了收藏功能、用户也不能对电影或电视剧进相应的评论等。

2.2.3 其他需求

时刻注意新片的上映。管理员应该时时刻刻注意电影的上映，一旦发现有新

的电影上映，管理员应该及时将它加进去。

时刻查看用户的评论。管理员应该时刻查看用户的评论，一旦发现有评论违反相对应的规定，管理员应及时删除该评论。

定期查看用户。管理员应该定期查看用户的状态，一旦该用户违反某一规定应该及时进行删除该用户。

3 总体设计

3.1 运行环境

3.1.1 配置环境

Jdk 1.8、tomcat 9.0.14、MySQL 8.0.14、IDEA 2019.1.2

使用这些配置环境，使得开发更方便，并且许多 jar 包不需要自己引入，只需要在配置 jdk 就可以；并且将系统在 tomcat 上运行，使得开发更方便；使用 MySQL 数据库能够更好的对数据进行存储和操作；在 IDEA 上创建项目写代码，使得开发人员开发更便捷。

3.1.2 框架与相应插件环境

Spring 5.1.8、spring MVC 5.1.8、mybatis 3.5.2、pagehelper 5.1.9

使用这些框架及插件，使得开发更迅速；一整套的框架让聚合性更小，对于维护更方便；使用相应插件使得开发某一功能更容易。

3.1.3 操作系统

64 位 win10

3.2 基本处理流程

根据这个网站的设计，主要分成前台和后台两个部分，这两个部分的流程示意图如下。

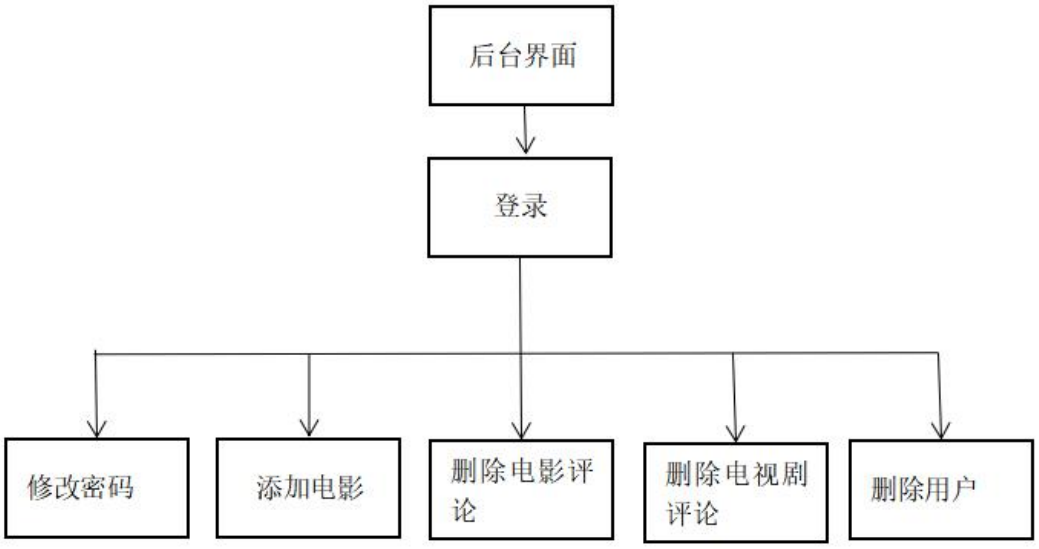

图 3-1 后台业务逻辑图

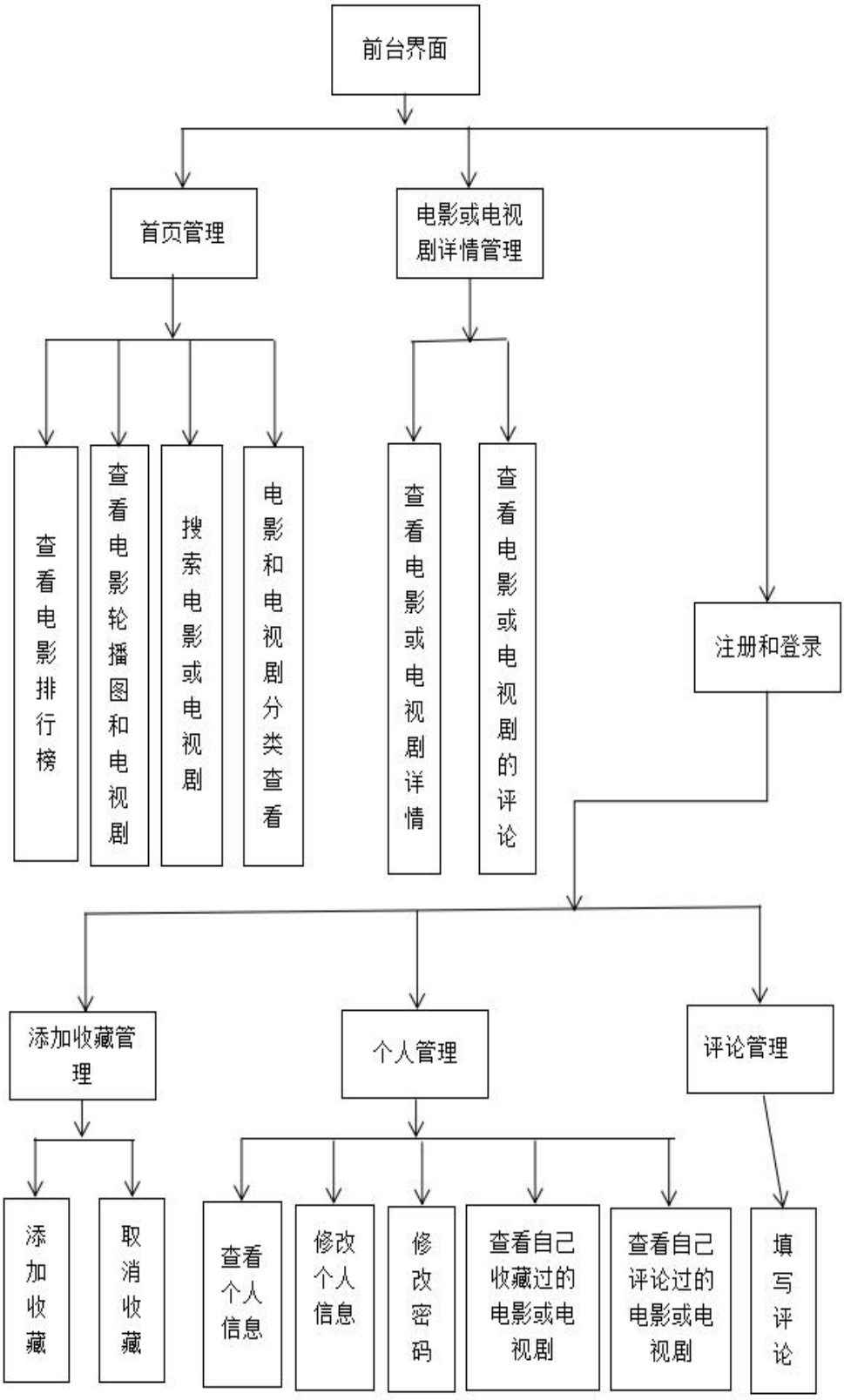
前台界面
首页管理
电影或电视剧详情管理
注册和登录
查看电影排行榜
查看电影轮播图和电视剧
搜索电影或电视剧
电影和电视剧分类查看
查看电影或电视剧详情
查看电影或电视剧的评论
添加收藏管理
个人管理
评论管理
添加收藏
取消收藏
查看个人信息
修改个人信息
修改密码
查看自己收藏过的电影或电视剧
查看自己评论过的电影或电视剧
填写评论

图 3-2 前台业务逻辑图

3.3 模块结构

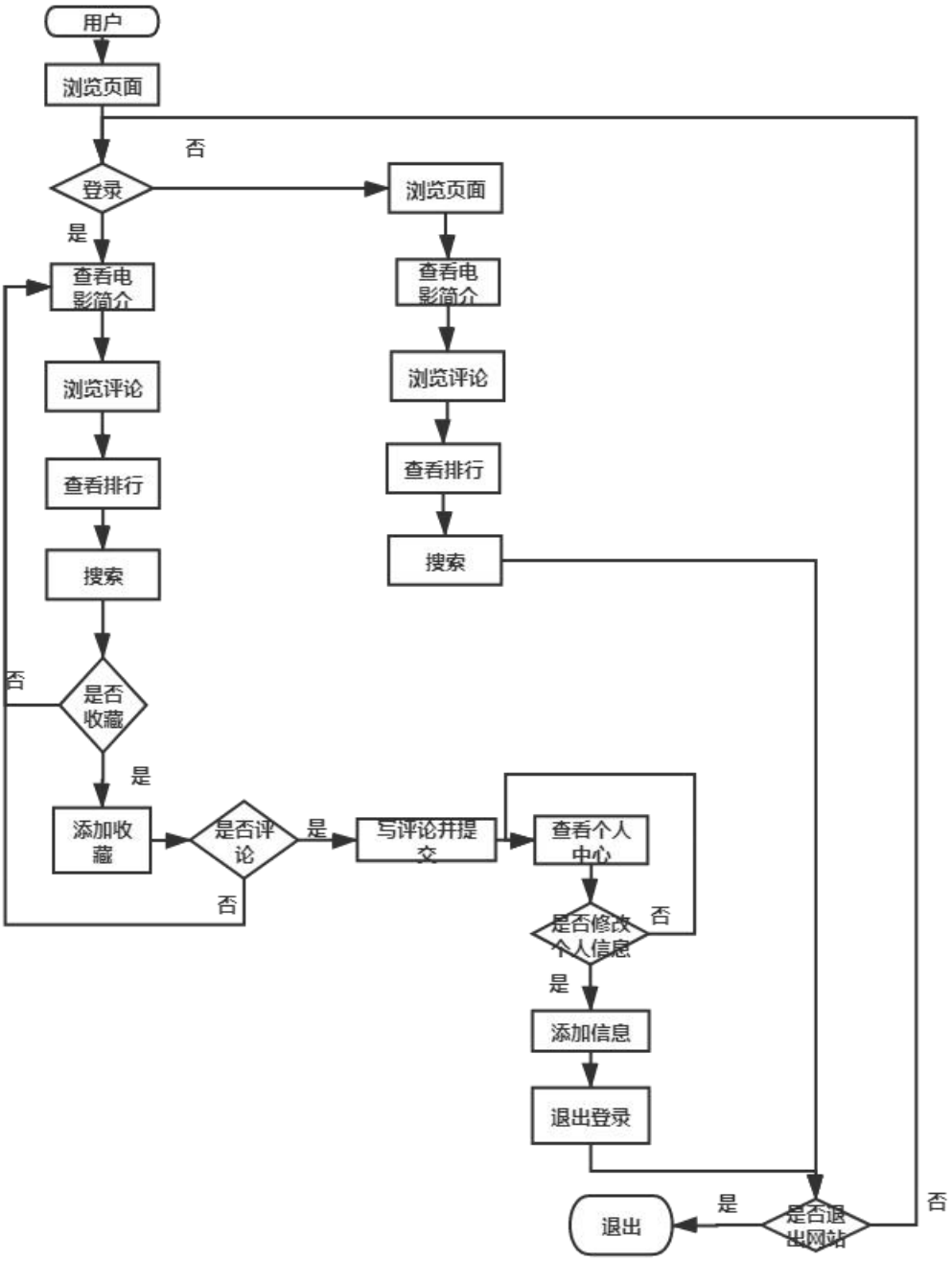

图 3-3 网站总功能流程图

3.3.1 电影展示和电视剧展示

电影展示和电视剧展示模块主要包含电影展示电影展示、电视剧展示、评分前十排行榜、搜索电影和搜索电视剧。电影展示和电视剧展示都分为几类：首页以轮播图的展示、全部电影和全部电视剧以列、半平铺、全平铺的方式展示。评分前十排行榜是通过项目提供的对这些电影的评分进行一个排名，并将前面十名展示出来。搜索电影和搜索电视剧是通过搜索栏搜索自己想要的电影，该功能是

通过模糊查询实现的。

3.3.2 电影分类和电视剧分类

电影分类和电视剧分类是将这些电影电视剧分为大致的以下七类：剧情、动画、爱情、奇幻、犯罪、惊悚、动作。用户通过挑选自己喜欢的类型，即可找到相应的电影和电视剧。

3.3.3 电影详情和电视剧详情

电影详情和电视剧详情主要包含以下信息：电影或电视剧封面，电影或电视剧名称，电影或电视剧总评分、电影或电视剧概述、电影或电视剧主演名单、电影或电视剧评论总条数，该电影或电视剧的所有评论、对电影或电视剧进行添加到最喜爱的里面去。

3.3.4 电影评论和电视剧评论

电影评论和电视剧评论是在电影详情页进入的，进行评论需要先进行登录。评论主要分为三个部分：标题、星级评分、内容。当评论提交之后就会在该页面回显出来，回显出来的内容包含标题、星级评分、内容、评论时间、评论作者等。

3.3.5 个人信息管理

个人信息管理是需要在登录之后才可通过点击用户名进去的，该模块主要包含：个人信息、修改个人信息、修改密码、查看自己所评论的电影和电视剧、查看自己所收藏的电影和电视剧。个人信息主要包含用户名、邮箱、名字、姓、家庭地址；然而修改信息只能修改名字、姓、家庭地址；修改密码则需要输入两次密码进行确认，确认通过即可；查看自己所评论的电影和电视剧可以看到自己所评论过的所有电影和都电视剧，同时可以看到自己所评论的内容等；查看自己所收藏的电影和电视剧可以看到自己所收藏过的所有电影和都电视剧，也可点击电影或电视剧进入相应的电影或电视剧详情页。

3.3.6 后台管理

后台管理模块主要包含修改管理员密码、添加电影、删除电影评论、删除电视剧评论、删除用户。修改管理员密码首先需要确认两次密码，确认成功即可修改成功；删除电影评论则是管理员删除一些不怎么符合规定的电影评论；删除电视剧评论则是管理员删除一些不怎么符合规定的电视剧评论；删除用户则是管理员删除一些已弃用或者违反规定的账号；添加电影则是管理员根据电影新片添加一些新上映的电影，主要需填写电影名字、电影封面、电影类别、电影简介、电影评分字段，最后点击提交即可添加进去。

3.4 外部接口

外部接口，一般在注册的时候需要发送短信验证码，那么就需要调用第三方短信接口，而且一般正式系统或网站需要的上线的话一般都需要调用外部接口。当项目需要完善时，就应该在注册时调用短信接口，如果想更加升级时，可在里

面添加电影订票功能，那么就需要调用第三方支付接口。

3.5 内部接口

在项目中，一般把代码分成 dao、pojo、service、controller 这几个部分，各个部分的功能不一样。pojo 主要是实体类；dao 层是与数据层连接的，service 是对一些事务进行处理，即逻辑层；controller 层主要调用 service 层进行实现功能。

这个网站对功能分析有如下内部接口。首先是 AdminService 接口，该接口主要是用于删除管理员、添加管理员、查询一个管理员、查询所有管理员、修改管理员等；UserService，该接口主要是用于删除用户、添加用户、查询一个用户、查询所有用户、修改用户等；UserInfoService,该接口主要是用于删除用户信息、添加用户信息、查询一个用户信息、查询所有用户信息、修改用户信息等；MovieService，该接口主要是用于删除电影、添加电影、查询一个电影、查询所有电影、修改电影等；MovieCommentService，该接口主要是用于删除电影评论、添加电影评论、查询一个电影评论、查询所有电影评论、修改电影评论等；TVPlayService，该接口主要是用于删除电视剧、添加电视剧、查询一个电视剧、查询所有电视剧、修改电视剧等；TVPlayCommentService，该接口主要是用于删除电视剧评论、添加电视剧评论、查询一个电视剧评论、查询所有电视剧评论、修改电视剧评论等；FavoriteService，该接口主要是用于删除收藏、添加收藏、查询一个收藏、查询所有收藏、修改收藏等。

4 数据库设计

4.1 概念结构设计

4.1.1 设计思路

做一个网站或者系统的时候，最主要的就是满足客户的需求。那么在做这个网站的设计思路是：第一步，把自己想象成使用这个网站的用户，细丝慢酌自己想要干什么，又想希望这个网站能有什么功能，当把一系列的功能想好，就要考虑实现这个功能有需要什么字段，逐步考虑，最后将这些字段统计起来，进行分类，每一类建一张表，最后再考虑一下这些表之间应该有什么关系，统计起来，然后建表的时候按照自己策划的来建就可以。

开发本电影评论网站，需要实现评论、查看电影、查看电影简介、用户注册登录、电影收藏、后台管理等功能。逐步分析，循序渐进的完成。

4.1.2 E-R 图

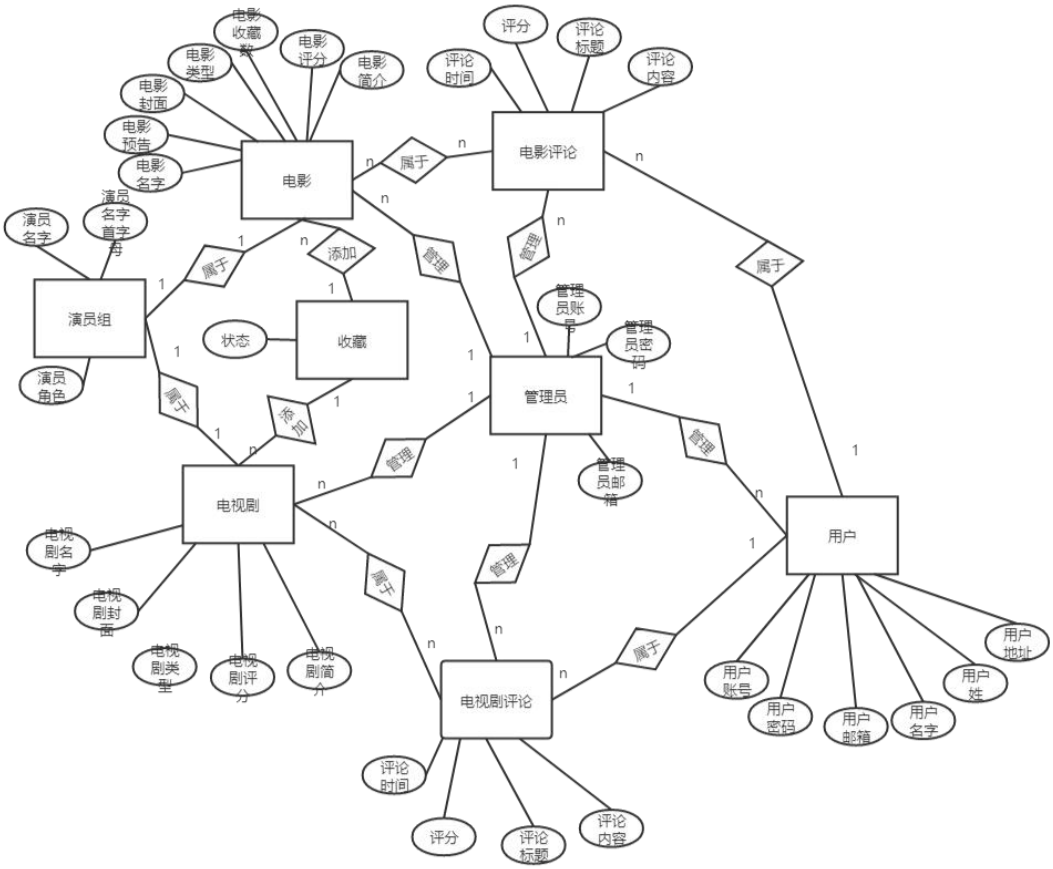

图 4-1 系统 E-R 图

4.2 逻辑结构设计

4.2.1 设计思路

通过上面的 E-R 土就可以清晰的知道各个实体的关系和他们的属性，先按照实体和属性建表出来，再考虑好这些表之间的关系，这样，数据库的建造就完成。

4.2.2 逻辑模型

根据实体与属性的关系和实体之间的关系同时从实际出发得到如下各表：

管理员信息表主要是记录管理员的账号和密码等。

表 4-1 管理员信息表 admin

字段名字	数据类型	长度	小数点	允许空值	备注
adminId	int	20	0	否	管理员 ID
adminName	varchar	255	0	是	管理员账号
adminPassword	varchar	255	0	是	管理员密码
adminEmail	varchar	255	0	是	管理员邮箱

演员信息表主要统计了所有演员的信息，其信息包括演员名字、身份等。

表 4-2 演员信息表 castcrew

字段名字	数据类型	长度	小数点	允许空值	备注
castCrewId	int	20	0	否	演员组 ID
name	varchar	50	0	是	演员名字
nameInitials	varchar	10	0	是	演员名字首字母
role	varchar	255	0	是	演员身份

电影信息表的作用主要是收集了所有电影的信息，主要信息为电影名字、电影封面图 URI、电影概述、电影类型、电影评分等。

表 4-3 电影信息表 movie

字段名字	数据类型	长度	小数点	允许空值	备注
movieId	int	20	0	否	电影 ID
movieName	varchar	255	0	是	电影名字
movieCover	varchar	255	0	是	电影预告 URL

续表 4-3

字段名字	数据类型	长度	小数点	允许空值	备注
movieUri	varchar	255	0	是	电影封面 uri
movieType	varchar	50	0	是	电影类型
movieFavoriteNum	int	255	0	是	电影被收藏数
movieStar	double	5	1	是	电影评分
movieOverview	varchar	255	0	是	电影简述
movieReviewId	int	20	0	是	电影评论 ID
movieCastCrewId	int	20	0	是	电影演员组 ID
userId	int	20	0	是	用户 ID

电影评论表的作用主要是收集了所有所有用户给电影的评论，该评论主要包含了用户的唯一标识 ID、电影的唯一标识 ID、评论时间、评分、评论标题、评论内容等。

表 4-4 电影评论表 moviecomment

字段名字	数据类型	长度	小数点	允许空值	备注
movieCommentId	int	255	0	否	电影评论 ID
userId	int	20	0	否	用户 ID
movieId	int	20	0	是	电影 ID
commentTime	varchar	255	0	是	评论时间
commentStar	int	5	0	是	评分
commentTitle	varchar	255	0	是	评论标题
commentContent	varchar	255	0	是	评论内容

电视剧信息表的作用主要是收集了所有电视剧的信息，主要信息为电视剧名字、电视剧封面图 URI、电视剧概述、电视剧类型、电视剧评分等。该表主要用来电视剧的展示。

表 4-5 电视剧信息表 tvplay

字段名字	数据类型	长度	小数点	允许空值	备注
TVId	int	20	0	否	电视剧 ID
TVName	varchar	255	0	是	电视剧名字
TVUri	varchar	255	0	是	电视剧预告
TVType	varchar	255	0	是	电视剧类型
TVStar	double	5	1	是	电视剧评分
TVOverview	varchar	255	0	是	电视剧概述

电视剧评论表的作用主要是收集了所有用户给电视剧的评论，该评论主要包含了用户的唯一标识 ID、电视剧的唯一标识 ID、评论时间、评分、评论标题、评论内容等。

表 4-6 电视剧评论表 tvplaycomment

字段名字	数据类型	长度	小数点	允许空值	备注
typlayCommentId	int	255	0	否	电视剧评论 ID
userId	int	20	0	否	用户 ID
typlayId	int	20	0	是	电视剧 ID
commentTime	varchar	255	0	是	评论时间
commentStar	int	5	0	是	评分
commentTitle	varchar	255	0	是	评论标题
commentContent	varchar	255	0	是	评论内容

用户账号表主要是注册和登录的时候使用，注册的时候需要往该表里面添加数据，登录的时候需要对该表进行查询。该表里面主要是维护了用户的账号、用户的密码以及用户的邮箱等。

表 4-7 用户账号表 user

字段名字	数据类型	长度	小数点	允许空值	备注
userid	int	10	0	否	用户 ID
username	varchar	20	0	是	用户账号
password	varchar	30	0	是	用户密码
email	varchar	30	0	是	用户邮箱

用户信息表主要的作用是统计所有用户的信息，并且用户可在个人信息管理

里面修改信息，主要的信息为用户的姓氏、用户的名字、用户的居住地址等。

表 4-8 用户信息表 userinfo

字段名字	数据类型	长度	小数点	允许空值	备注
userInfoId	int	20	0	否	用户信息 ID
userId	int	20	0	是	用户 ID
firstName	varchar	255	0	是	用户名字
lastName	varchar	255	0	是	用户姓氏
address	varchar	255	0	是	用户地址
avatar	varchar	255	0	是	

收藏信息表的作用主要是收集了所有用户收藏的电影和电视剧，该表主要包含了用户的唯一标识 ID、电影的唯一标识 ID、电视剧的唯一标识 ID、收藏的状态等。

表 4-9 收藏信息表 favorite

字段名字	数据类型	长度	小数点	允许空值	备注
favoriteId	int	20	0	否	收藏 ID
userId	int	20	0	是	用户 ID
movieId	int	20	0	是	电影 ID
tvplayId	int	20	0	是	电视剧 ID
status	int	1	0	是	收藏状态

4.3 物理结构设计

4.3.1 存取方式

存取方式是用户访问数据库的时候会通过什么方式去访问，然而不同的方式对应不同的效率。一般数据库里面，通常会通过建立索引来存取数据。在这里，使用的存储引擎是 innodb 存储引擎。然而为了拥有更快的存取速度，这里采用的存取方式是快速数据库存取，使用这种方式可以快速的对数据库进行读取，因此，这里就使用快速数据库存取的存取方式。

4.3.2 存储结构

这个网站，采用的是 MySQL 数据库，MySQL 拥有两种存储结构，分别是 B 树存储结构，hash 存储结构。一般大部分人都喜欢使用 B 数存储结构，当然，也不例外，在这里，使用的就是 B 树存储结构。

人们之所以喜欢使用 B 树存储结构就是因为 mysql 数据库的数据是放到外部存储的,因此我们必须减低访问内存的 IO 次数,所以我们尽能力把树的高度降

低，所以需要做到这一点，必须把树的分叉弄得越多越好，因此，B 树就符合我们的需求。

5 界面设计

5.1 界面关系图或工作流图

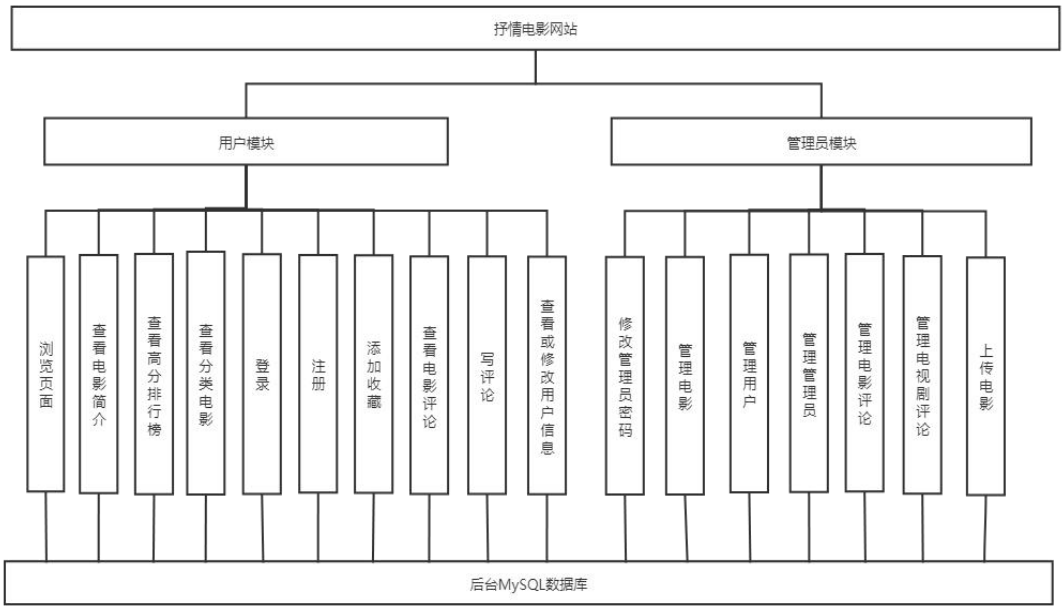

图 5-1 系统功能图

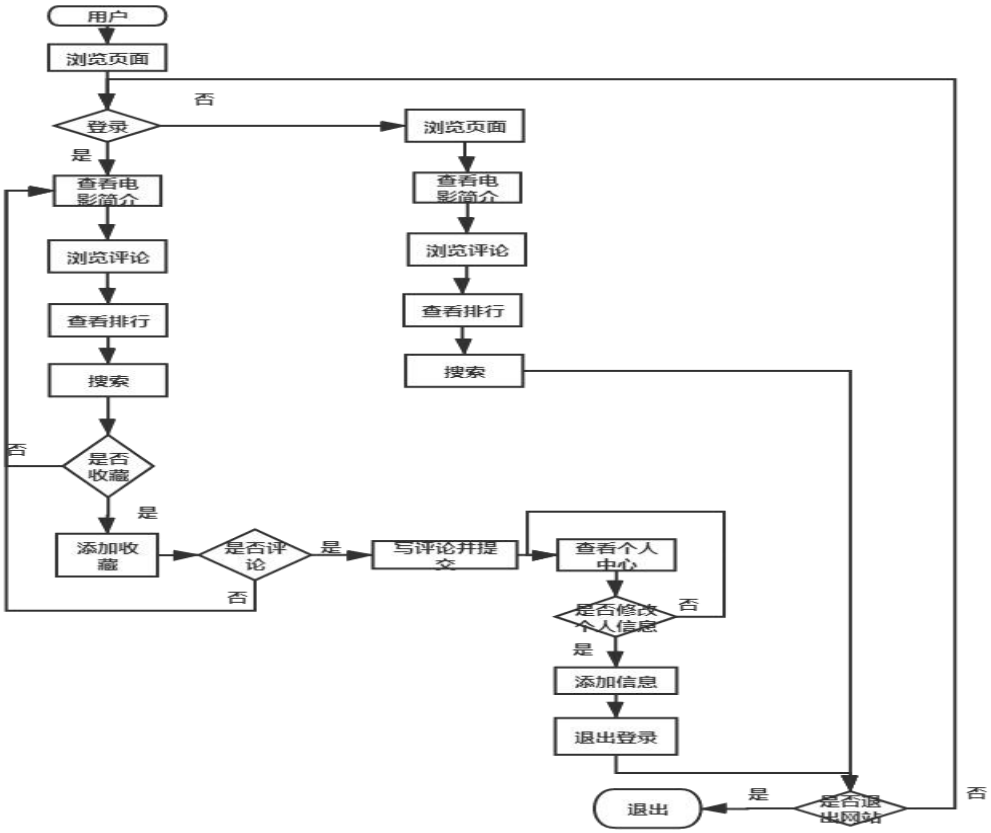

图 5-2 工作流图

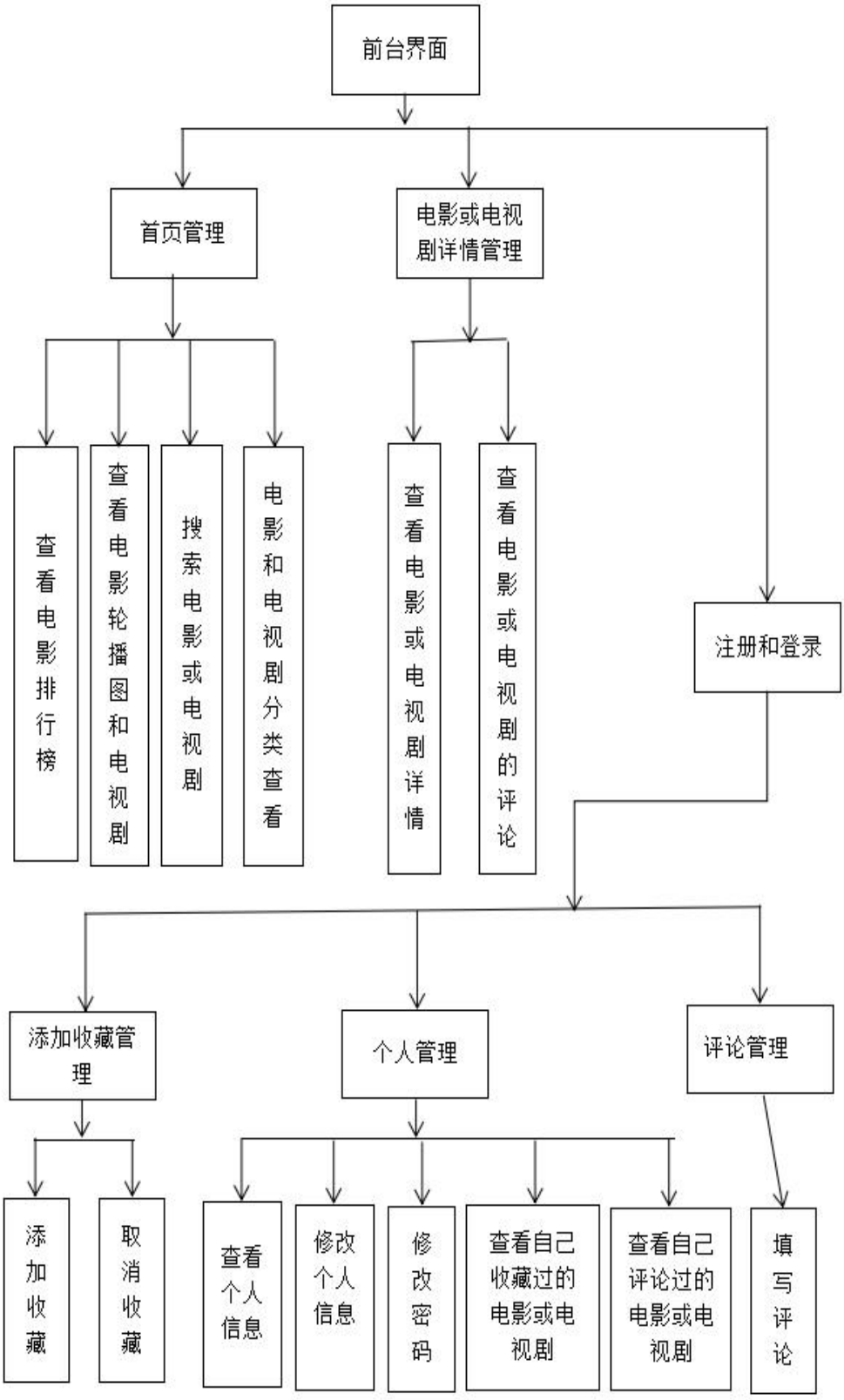
前台界面
首页管理
电影或电视剧详情管理
注册和登录
查看电影排行榜
查看电影轮播图和电视剧
搜索电影或电视剧
电影和电视剧分类查看
查看电影或电视剧详情
查看电影或电视剧的评论
添加收藏管理
个人管理
评论管理
添加收藏
取消收藏
查看个人信息
修改个人信息
修改密码
查看自己收藏过的电影或电视剧
查看自己评论过的电影或电视剧
填写评论

图 5-3 界面关系图

用户当未登录的时候，只能进行查看电影和电视剧等，只有登录之后，用户才能进行评论、收藏电影或者电视剧，并且还能维护个人信息。

5.2 界面设计成果

项目中的所有请求都是通过 ajax 发送的。

5.2.1 主界面

(1) 首页

首先，先来介绍一下的首页，首页主要是排行榜和轮播图以及推荐电影和电视剧，还有处于顶部的导航栏以及搜索栏。顶部导航栏是同 ul 标签完成，搜索栏是通过下拉选择电影或电视剧，再输入内容进行模糊查询。轮播图是通过监听事件完成，然而排行榜是通过查询数据库的评分字段进行排序及挑出最高的十名。当然，还有所有的电影和电视剧页面，这些都是采用分页处理。

如图 5-4 所示：

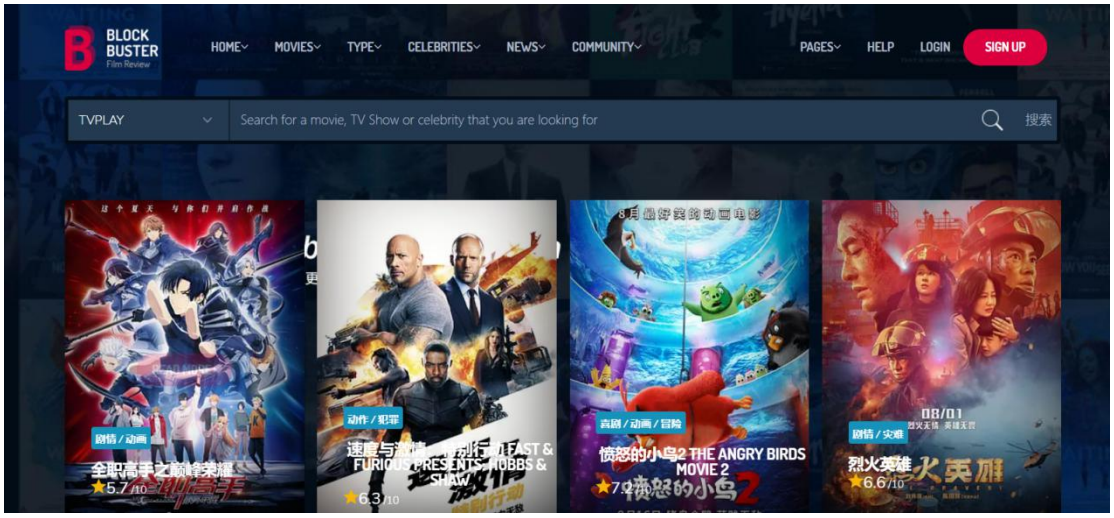

图 5-4 首页效果图

(2) 登录

登录页面则是采用一个模态框，当点击时就会出现。需输入用户名、用户密码、选择 role。这个界面挺简单，但是当初必输入的字段为空时，页面就会弹出提示。

如图 5-5 所示：

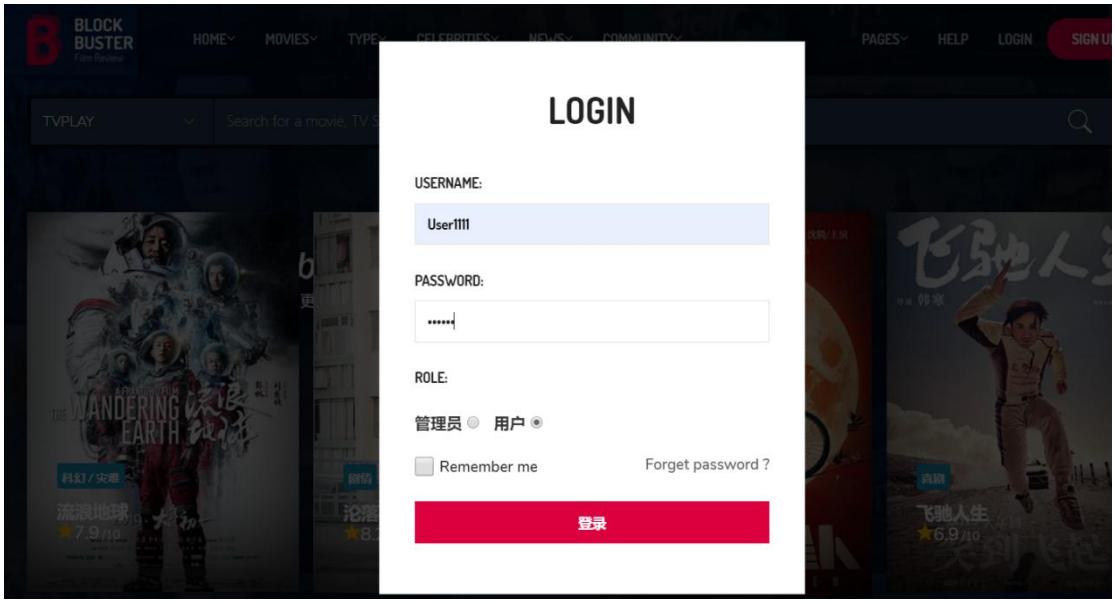

图 5-5 登录效果图

(3) 注册

注册页面其实与登录页面使用的方法一致，都是采用模态框，需输入用户名、用户邮箱、用户密码、确认输入密码，并且当必输字段为空时，页面会弹出提示。

如图 5-6 所示：

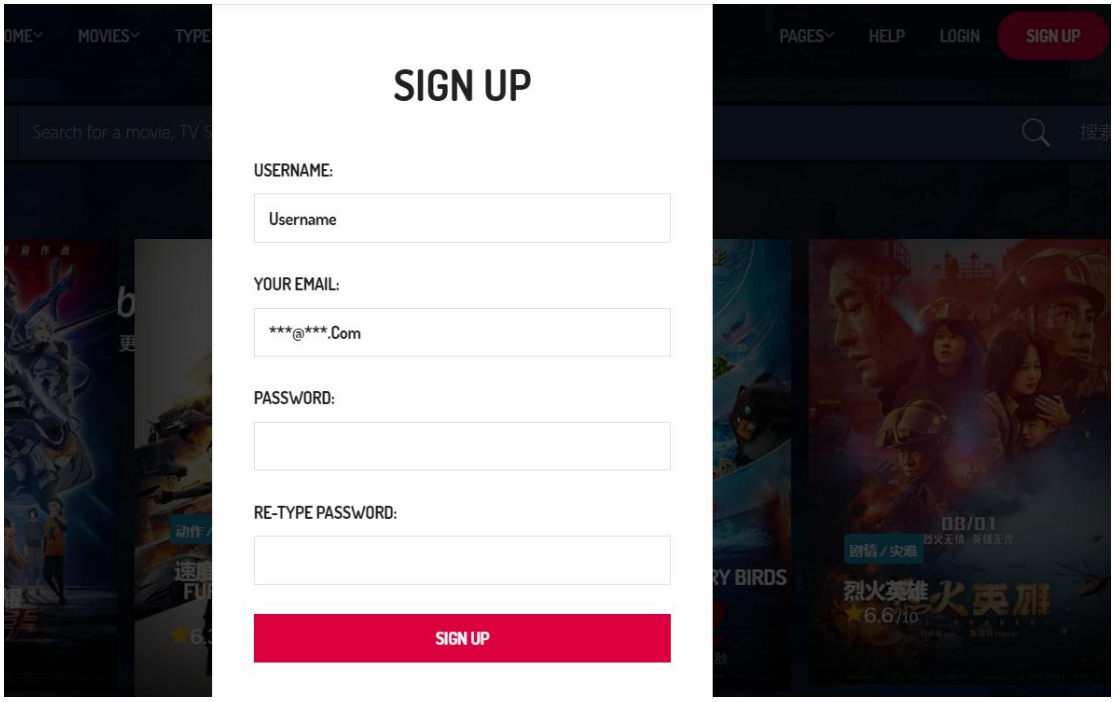

图 5-6 注册效果图

(4) 电影或电视剧详情页电影或电视剧详情页面主要左侧是一张电影封面，右边拥有这些内容：电影或电视剧标题、收藏按钮、电影或电视剧评分、电影或电视剧概述、电影或电视剧、电影或电视剧评论。这些数据都是通过电影或者电视剧

id 到数据库里面查询出的数据。

如图 5-7 所示：

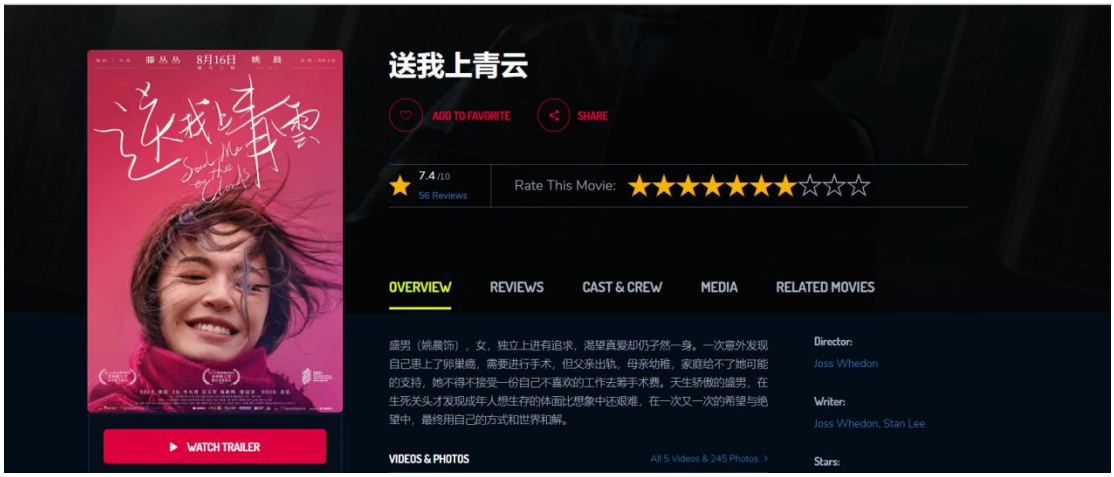

图 5-7 电影或电视剧详情效果图

(5) 收藏页

收藏页其实就是在电影或电视剧详情页面中的一个按钮，当点击时就会收藏，样式也就会改变，再次点击就会取消收藏，同样的样式也会随之改变。

如图 5-8 所示：

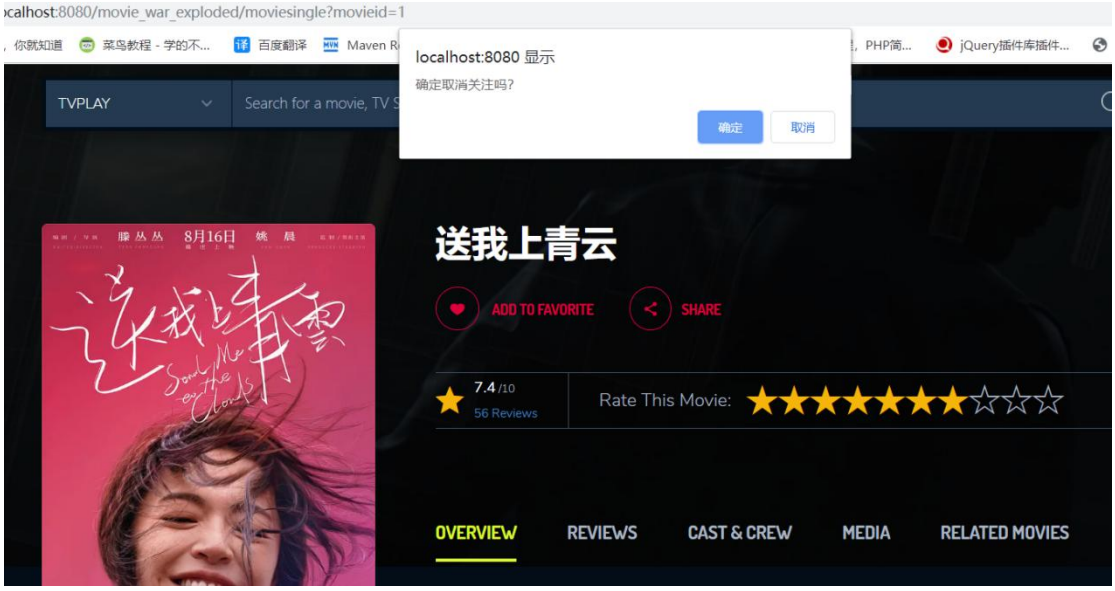

图 5-8 收藏页效果图

(6) 添加评论页

添加评论页其实就是在电影或电视剧详情页面中有个导航栏，当选择评论时，评论的右上部分就会出现写评论的按钮，当点击之时，该页面就会弹出一个模态框，里面需要输入评论标题、评论评分、评论内容方可提交。

如图 5-9 所示：

图 5-9 添加评论页效果图

(7) 用户信息页

用户信息页则是已经登录后导航栏上点击用户名即可进入，信息页面不能修改信息，只可查看信息，主要有用户名、邮箱、姓、名字、地址等信息。

如图 5-10 所示：

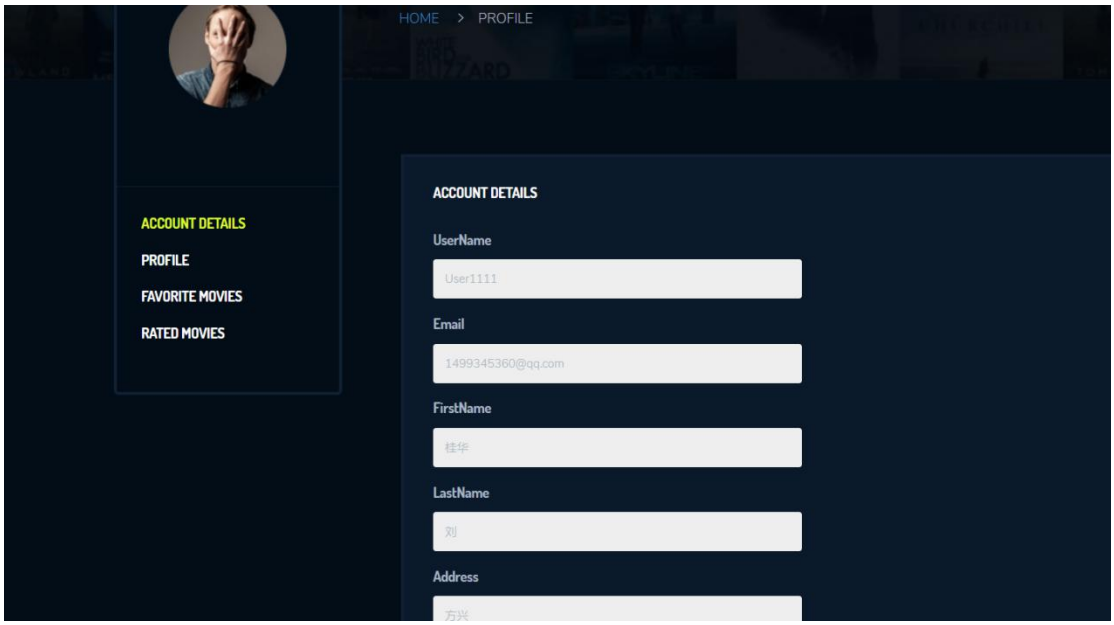

图 5-10 用户信息页效果图

(8) 用户修改信息页

用户修改信息页则是已经登录后导航栏上点击用户名即可进入，修改信息页面只提供修改信息，主要有修改基本信息和修改密码等信息。修改基本信息需要填入修改的字段并保存，修改密码则是需要输入正确密码并提交。该一系列的请求都是通过 ajax 来发送的。

如图 5-11 所示：

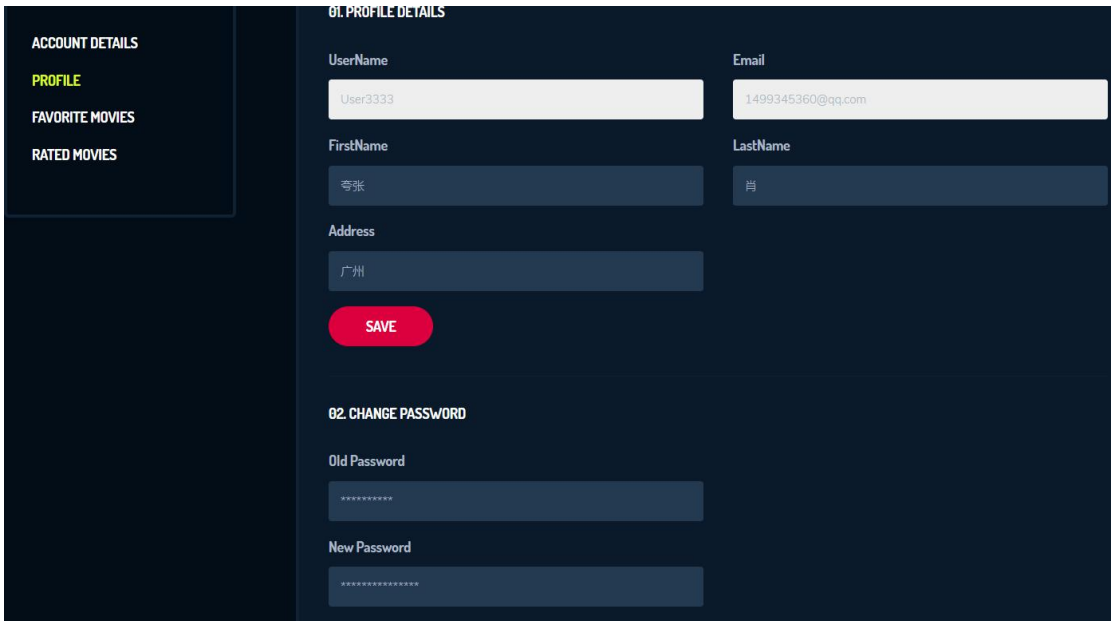

图 5-11 用户修改信息页效果图

(9) 用户添加收藏列表页

用户添加收藏列表页则是已经登录后导航栏上点击用户名即可进入，用户添加收藏列表页面是通过用户名查找已经收藏的电影或电视剧，并将它们以分页的形式展示出来。

如图 5-12 所示：

图 5-12 用户添加收藏列表页效果图

(10) 用户已评论列表页

用户已评论列表页则是已经登录后导航栏上点击用户名即可进入，用户已评论列表页面是通过用户名查找已经评论的电影或电视剧，并将它们以分页的形式展示出来。其实大致与收藏列表类似。

如图 5-13 所示：

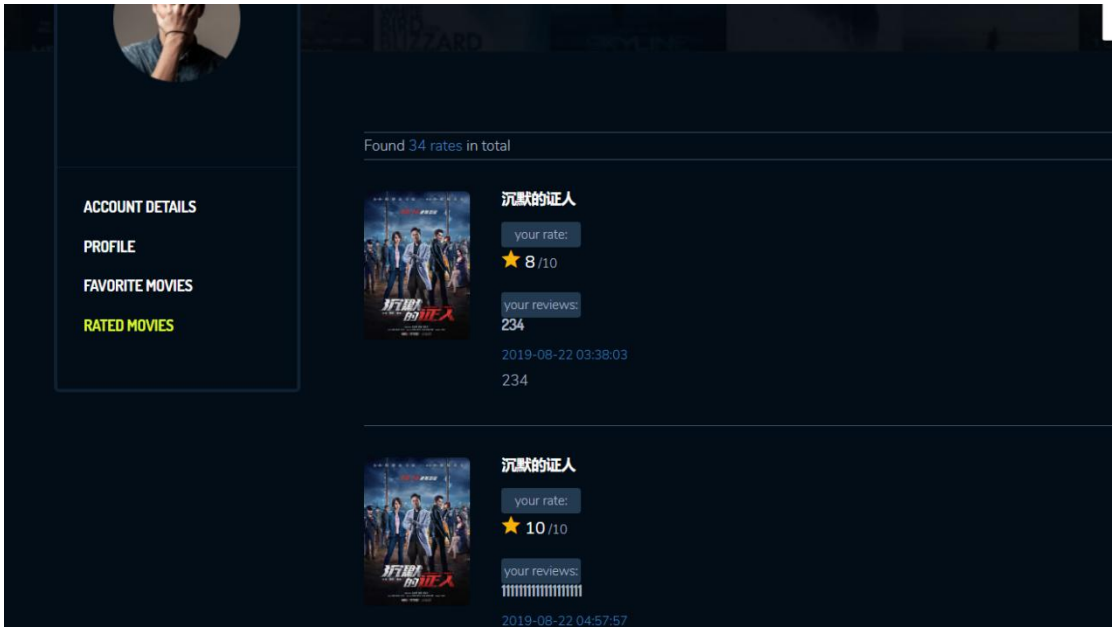

图 5-13 用户已评论列表页效果图

(11) 后台管理修改密码页

后台管理修改密码页则是已经登录管理员后的页面，该页面需要填充以下字段，管理员原始密码、新密码、确认新密码，只有填充好这些字段，方可修改密码。

如图 5-14 所示：

图 5-14 后台管理修改密码页效果图

(12) 后台管理列表页

后台管理列表页则是已经登录管理员后的页面，不同页面有不同的列表，这

里只展示电影列表，电影列表展示的是数据库中所有电影，主要展示封面、概述、名字、类型、评分等信息，管理员可以等它们进行管理。

如图 5-15 所示：

图 5-15 后台管理列表页效果图

(13) 后台添加电影页

后台添加电影页是需要添加电影到数据库中，需要的字段为电影名字、电影封面、电影类别、电影简介、电影评分等，最后点击提交即可添加。

如图 5-16 所示：

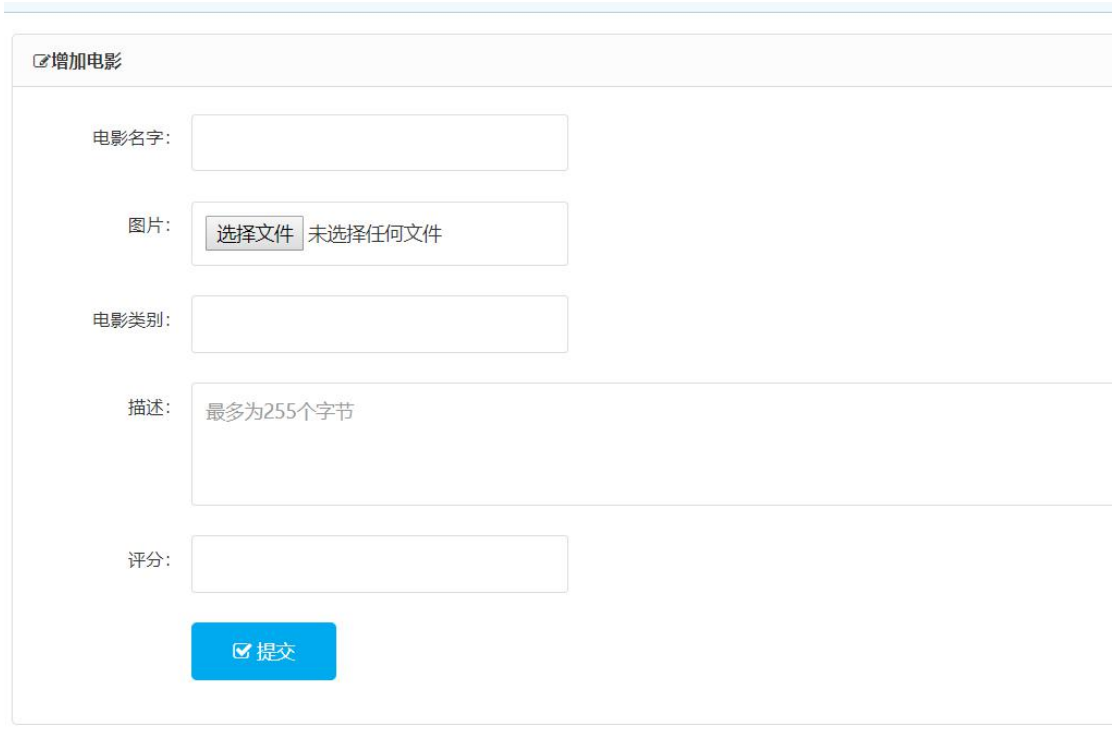

图 5-16 后台添加电影页效果图

6 详细设计

6.1 系统主要功能模块介绍

6.1.1 电影和电视剧展示模块

电影和电视剧展示模块分成电影展示、电视剧展示、搜索模块、电影和电视剧分页、电影排行榜展示等五个小模块。电影展示就是把数据库中的电影按照一定的样式展示出来；电视剧展示就是把数据库中电视剧按照一定的样式展示出来，该模块其实与电影展示类似；搜索模块就是输入你想要搜索的电影或者电视剧，后端将该字段以模糊查询的方式查询数据库，将查出来的内容展示出来；电影和电视剧分页就是当电影或电视剧数量过多时，就会采用分页的方式展示出来；电影排行榜就是把数据库中的电影按照评分从高到低排序，跳出最高的十部电影并展示出来。

6.1.2 电影和电视剧分类模块

电影和电视剧分类模块主要是把电影和电视剧按照类型分成以下几类：剧情、动画、爱情、奇幻、犯罪、惊悚、动作。用户可随意挑选他喜欢的类别，后端将该字段以模糊查询的方式查询数据库，并将查询的数据展示出来。

6.1.3 电影和电视剧详情模块

电影和电视剧详情模块是当用户想了解一部电影或者电视剧，就可以点击进去查看电影或者电视剧详情。该详情是通过将电影或电视剧名字传到后端，查询电影或电视剧表，并将它们的信息按照一定的央视显示出来。主要显示电影或者电视剧封面、电影或者电视剧名字、电影或者电视剧简介、电影或者电视剧评分、电影或者电视剧总评论、该电影或者电视剧是否被当前用户收藏。

6.1.4 用户中心模块

用户中心模块主要分成登录或注册、查看用户信息、修改用户信息、修改密码、查看已收藏列表、查看已评论列表、添加收藏、添加评论这几个小模块。登录主要是通过用户输入的用户名和密码，通过后端查询比对，如果相符合就允许登录，反之则不允许登录；注册则是将必填的字段填入，后端将会把这些字段以一条新纪录插入到数据库的用户表中；查看用户信息则是用户登录之后可查，主要是通过用户名查询用户信息表并将查询到的数据展示出来；修改用户信息则是将一些信息进行修改，然后后端将修改的字段及时更新到数据库中；修改密码则是用户输入正确密码和输入新密码传到后端，后端先比对密码是否正确，正确之后再进行在数据库的用户表中更新密码；查看已收藏列表就是通过用户名后端进行查询数据库中的评论表，并将查到的数据一一展示出来，并且采用分页处理；查看已评论列表就是通过用户名后端进行查询数据库中的收藏表，并将查到的数据一一展示出来，并且采用分页处理，该部分其实与查看已收藏列表类型；添加

收藏则是登录之后在电影或者电视剧详情页面点击那个收藏按钮即可在后端对收藏表进行添加一条新记录，反之取消之时就是删除一条记录；添加评论就是用户登录之后在电影或者电视剧详情页面找到写评论这个按钮，点击之后就会弹出一个模态框，用户填入相应的信息之后，点击提交，后端就会在评论表中添加一条新的数据。

6.1.5 管理员模块

管理员模块主要分成修改管理员密码、电影列表、电影评论管理、电视剧评论管理、用户管理、管理员列表这几个小模块。修改管理员密码则是管理员进入修改密码页面，输入对应字段，后端先校验密码正确与否，正确则进行更新数据库中的管理员表；电影列表则是展示出数据库中的所有电影，并按照列表的形式展示；添加电影则是管理员根据电影新片添加一些新上映的电影，主要需填写电影名字、电影封面、电影类别、电影简介、电影评分字段，最后点击提交即可添加进去，后端就会将这些数据以一条新纪录插入电影表中；电影评论管理则是管理员将审核电影评论，一旦违规就会删除该电影评论，同时数据库也会对删除该电影评论的记录；电视剧评论管理则是管理员将审核电视剧评论，一旦违规就会删除该电视剧评论，同时数据库也会对删除该电视剧评论的记录；用户管理则是管理员会查看该用户是否做出违规行为或者该用户此账号已弃用，管理员就会删除该用户，删除同时数据库也会删除这个用户的记录；管理员列表主要是展示出所有管理的列表，方便管理员管理。

6.2 电影和电视剧展示模块设计

6.2.1 电影和电视剧展示模块算法描述

电影和电视剧展示模块分成电影展示、电视剧展示、搜索模块、电影和电视剧分页、电影排行榜展示等五个小模块。电影展示、电视剧展示和电影排行榜是通过进入网站的时候分别请求 MovieService、TVPlayService 等接口，将获取到的数据按照排版的样式展示出来；搜索模块是通过在搜索框里面输入文字，点击搜索就通过 Ajax 发送请求，请求通过模糊查询查找 movie 和 tvplay,并将查到的数据展示出来；电影和电视剧分页是当进需要分页的页面，就会发送请求到后端，请求相应的数据，并通过 PageHelper 插件将其数据按照一定的规则分页处理。

6.2.2 电影和电视剧展示模块程序流程图

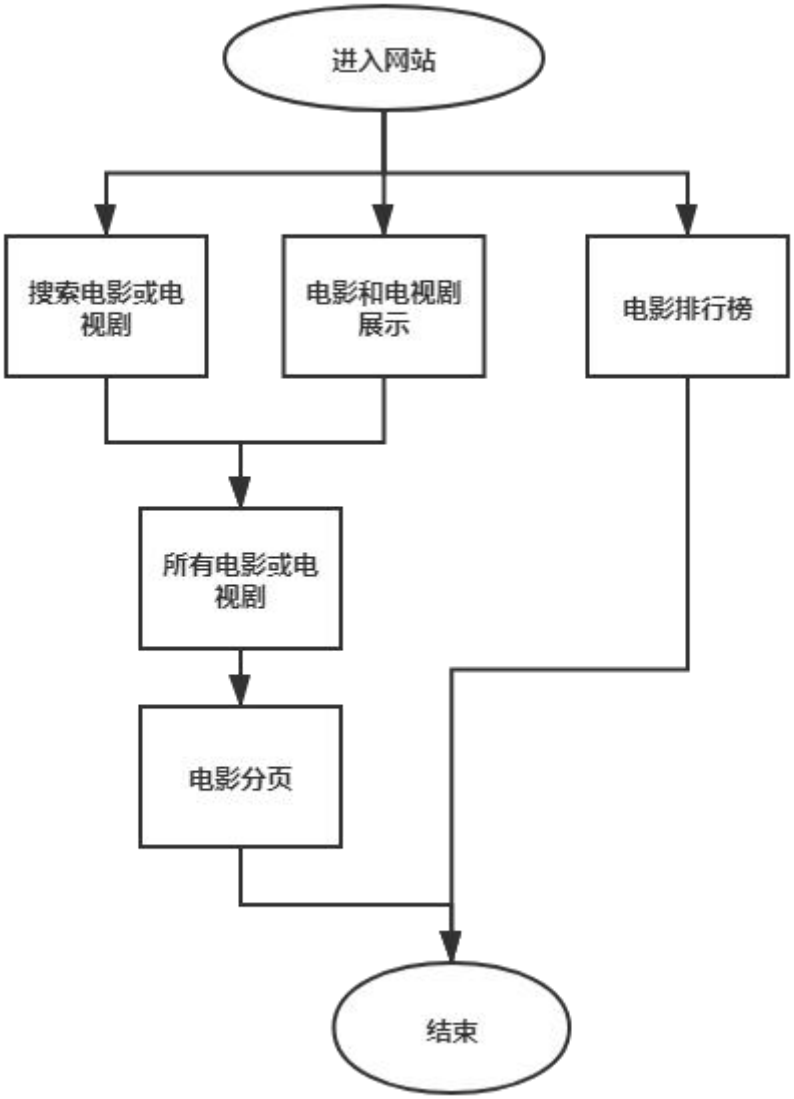

图 6-1 电影和电视剧展示模块程序流程图

6.2.3 电影和电视剧展示模块关键类说明

电影和电视剧展示模块主要是使用电影 Controller 类、电视剧 Controller 类。电影 Controller 类和电视剧 Controller 类都是通过接收 ajax 发送过来的数据，并通过数据进行一定的逻辑处理，最后将结果返回到前端页面，当需要进行分类的时候，就在逻辑处理过程中通过 PageHelper 对数据进行分页处理。电影 Controller 类和电视剧 Controller 类里面最主要的方法就是 public List<Movie> selectAll()方法，通过前端发送请求，然后后端以 json 数据的形式返回给前端。

6.3 电影和电视剧分类模块设计

6.3.1 电影和电视剧分类模块算法描述

电影和电视剧分类模块主要是通过用户选择相应的模块的同时，前端会通过Ajax发送一个请求到后端，后端通过这个类型去模糊查询数据库，并将查到的数据返回给前端页面。

6.3.2 电影和电视剧分类模块程序流程图

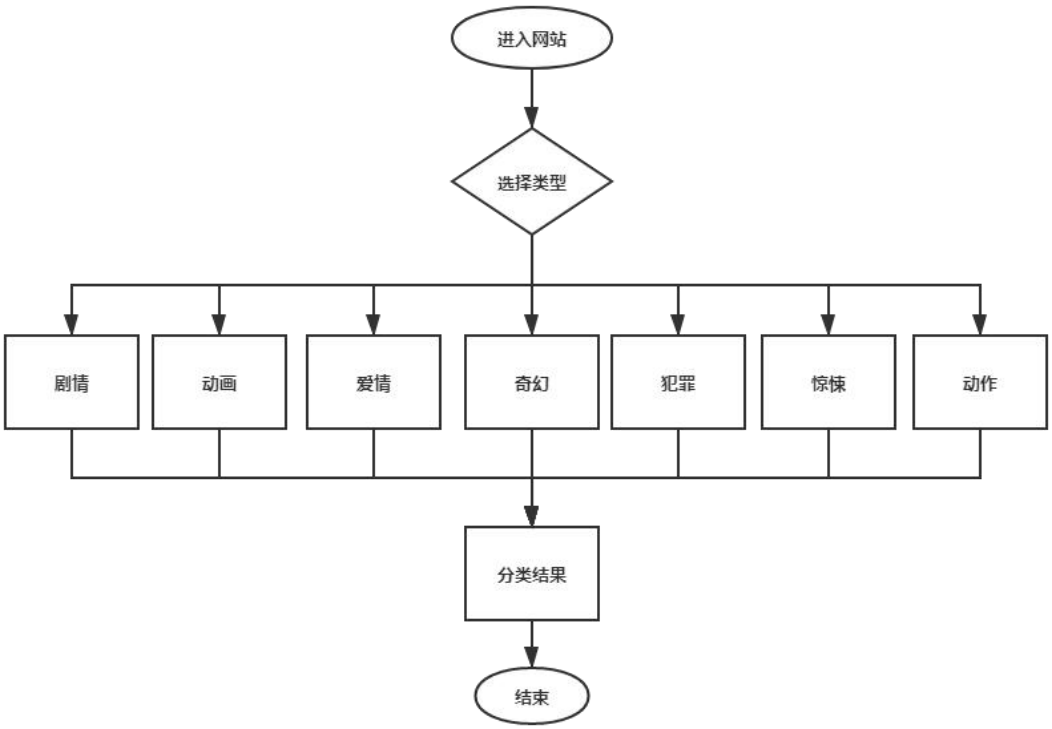

图 6-2 电影和电视剧分类模块程序流程图

6.3.3 电影和电视剧分类模块关键类说明

电影和电视剧分类模块主要是使用电影Controller类、电视剧Controller类。电影Controller类和电视剧Controller类通过接受到的前端发送过来的数据“类型”进行逻辑处理，处理完成后将会发送一个数据回去。该模块在电影Controller类和电视剧Controller类的主要方法是public List<Movie> searchByLike(String name)和public List<TVPlay> searchByLike(String name)，这两个方法通过前端传入所需要字段，后端查询数据库并返回结果。

6.4 电影和电视剧详情模块设计

6.4.1 电影和电视剧详情模块算法描述

电影和电视剧详情模块是用户随意点击一个电影或者电视剧的时候，前端就会将这个电影或者电视剧名字发送过去，同时，后端就会把这些数据作为关键词查询数据库表，并将当时单独的一个对象返回给前端。

6.4.2 电影和电视剧详情模块程序流程图

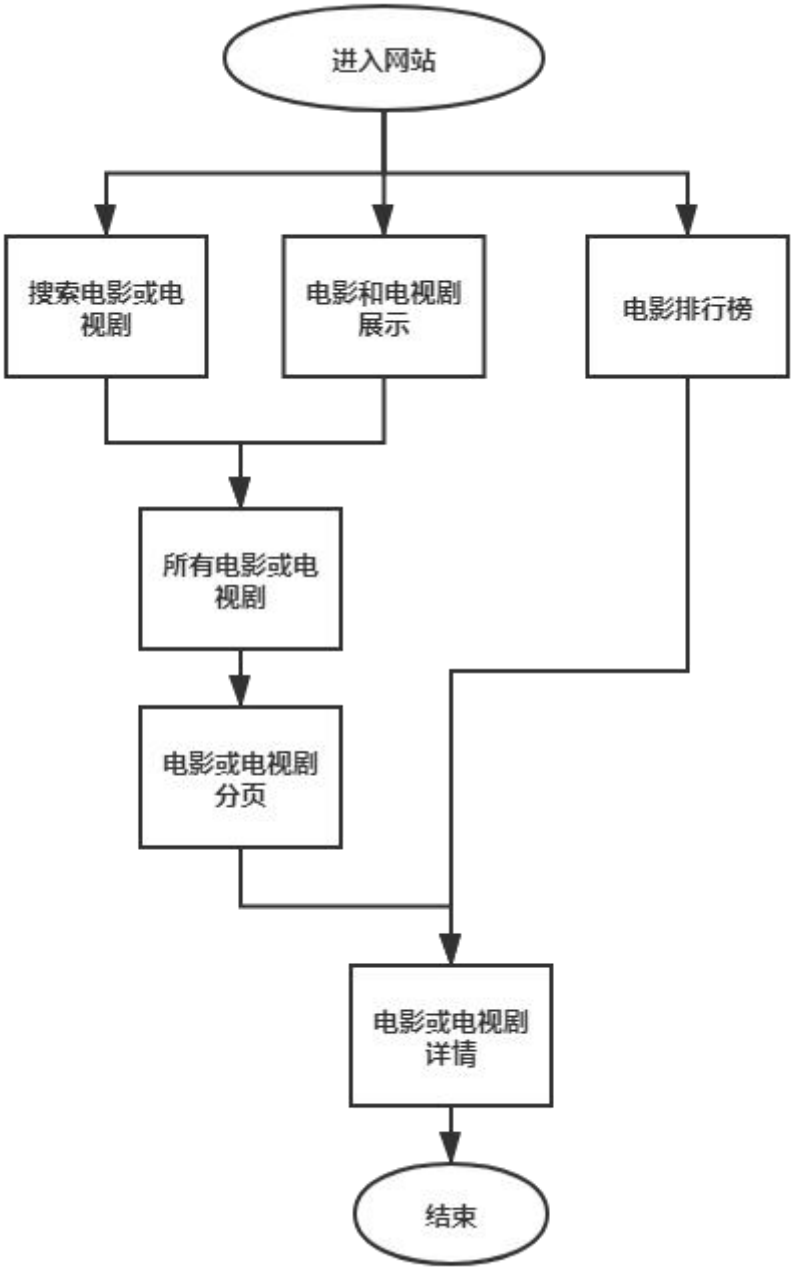

图 6-3 电影和电视剧详情模块程序流程图

6.4.3 电影和电视剧详情模块关键类说明

电影和电视剧详情模块主要使用电影 Controller 类、电视剧 Controller 类、电影评论 Controller 类、电视剧评论 Controller 类，这些接口都是对接收到的数据进行查询处理，并将数据返回到前端。该模块主要使用到的方法是 public TVPlay selectByPrimaryKey(Integer tvid) 和 public Movie selectByPrimaryKey(Integer movieid)，前端当你点击哪一个电影或电视剧时，后端就会根据所传字段查询该电影或电视剧的详细信息并返回给前端。

6.5 用户中心模块设计

6.5.1 用户中心模块算法描述

用户中心模块主要分成登录或注册、查看用户信息、修改用户信息、修改密码、查看已收藏列表、查看已评论列表、添加收藏、添加评论这几个小模块。用户注册和登录分别是对 user 数据表进行添加和查询操作；查看用户信息则是进入该页面的同时，对 userinfo 表进行查询，并将查到的一个对象返回前端；修改用户信息主要是通过提交修改发送 Ajax 请求，对 userinfo 表进行更新操作；修改密码则是通过提交修改发送 Ajax 请求，对 user 表进行更新操作；查看已收藏列表和查看已评论列表是通过用户名字查询 favorite 表、moviecomment 表和 tvplaycomment 表，将含有用户名的数据全部返回给前端页面；添加收藏是用户在详情页面发送一个对 favorite 表进行增加或删除的请求；添加评论是用户再提交评论的时候会发送一个添加评论的请求，发送这个请求之后，后端就会在 moviecomment 表或者 tvplaycomment 表添加一条记录。

6.5.2 用户中心模块程序流程图

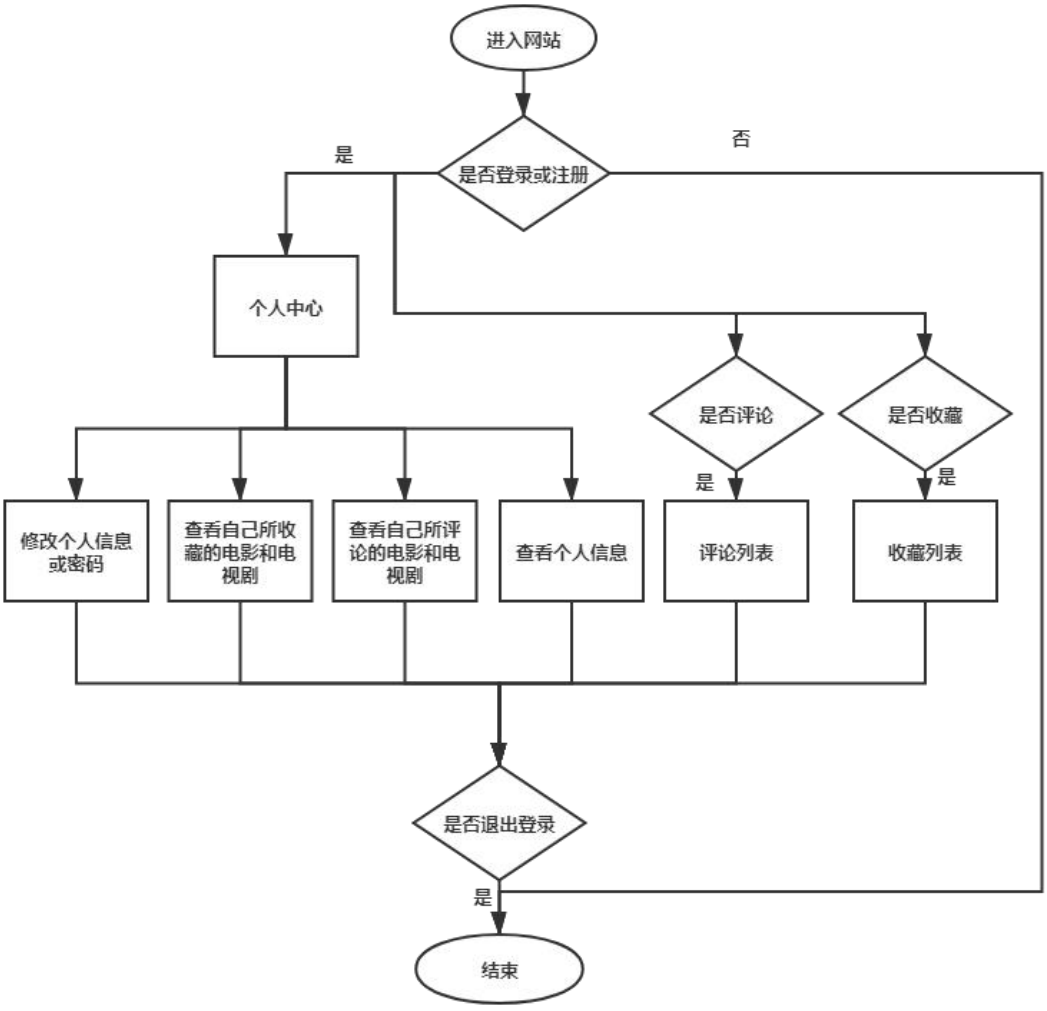

图 6-4 用户中心模块程序流程图

6.5.3 用户中心模块关键类说明

用户中心模块主要使用电影 Controller 类、电视剧 Controller 类、电影评论 Controller 类、电视剧评论 Controller 类、收藏 Controller 类电影评论 Controller 类、电视剧评论 Controller 类、用户信息 Controller 类。电影 Controller 类、电视剧 Controller 类在这个模块只进行查询操作；电影评论 Controller 类和电视剧评论 Controller 类在这个模块中进行的添加和查询操作；收藏 Controller 类在这个模块中进行添加或删除的操作；用户 Controller 类在这个模块中进行查询和增加操作、用户信息 Controller 类在这个模块进行查询、修改、增加等操作。主要操作方法有 public int updateByPrimaryKey(UserInfo record)、public UserInfo selectByUserId(Integer userid)等。

6.6 管理员模块设计

6.6.1 管理员模块算法描述

管理员模块主要分成修改管理员密码、电影列表、电影评论管理、电视剧评论管理、用户管理、管理员列表这几个小模块。修改管理员密码主要是通过提交修改发送请求，更新 admin 中的数据；电影列表主要是进入该页面就会发送请求查询到 movie 表中所有的数据；电影评论管理、电视剧评论管理、用户管理、管理员列表等都是进入该页面的同时会发送请求查询对应的数据，并按照对应的格式展示出来。同时，该模块中也会进行删除和修改操作，这个功能也就是对对应的数据表进行修改或者删除甚至添加等操作。

6.6.2 管理员模块程序流程图

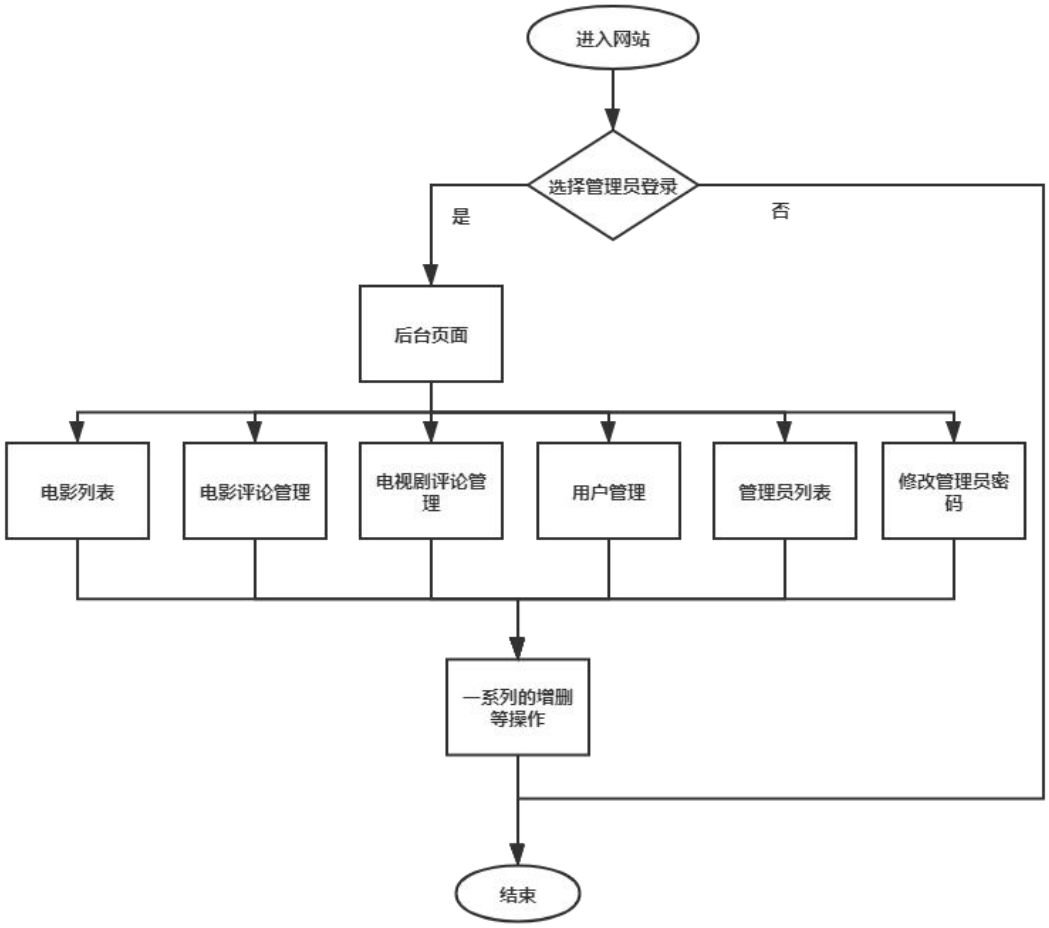

图 6-5 管理员模块程序流程图

6.6.3 管理员模块关键类说明

管理员模块主要使用电影 Controller 类、电影评论 Controller 类、电视剧评论Controller类、用户Controller类、管理员Controller类。电影Controller类主要是添加电影、删除电影和各种查询电影的方法，并且可以将查询的结果返回到前端页面；电影评论 Controller 类主要是添加电影评论、删除电影评论和各种查询电影评论的方法，并且可以将查询的结果返回到前端页面；电视剧评论Controller 类主要是添加电视剧评论、删除电视剧评论和各种电视剧电影评论的方法，并且可以将查询的结果返回到前端页面；管理员 Controller 类主要是查询和修改数据表，管理员账号是由系统所分发，人为不可注册的；用户Controller 类在这个模块中主要是体现查询和删除这一功能。该模块最主要的是上传电影，前端使用 form 表单形式，将数据传给后端，后端对这些数据进行校验，然后通过 public int insert(Movie record)该方法对数据库进行插入操作。

7 编码

7.1 代码实现与核心算法

```
@RequestMapping("/")
public String main(HttpServletRequest request){
    List<Movie> movieList=movieService.selectAll();
    request.getSession().setAttribute("movies",movieList);
    List<TVPlay> tvPlays=tvPlayService.selectAll();
    request.getSession().setAttribute("tvPlays",tvPlays);
    List<Movie> orders=movieService.searchByOrder();
    request.getSession().setAttribute("orders",orders);
    return "index1";
}
```

```
@RequestMapping("/search")
public String search(@RequestParam(value = "search")
String search, @RequestParam(required = false,
value="pn",defaultValue="1")Integer pn,
HttpServletRequest request){
    PageHelper.startPage(pn, 5);
    String table=request.getParameter("show_type");
    request.setAttribute("table",table);
    if ("Movie".equals(table)){
        List<Movie> movies=movieService.searchByLike(search);
        PageInfo pageInfoMovies = new PageInfo(movies,3);
        request.setAttribute("pageInfo", pageInfoMovies);
    }else if ("TVPlay".equals(table)){
        List<TVPlay> tvPlays=tvPlayService.searchByLike(search);
        PageInfo pageInfoTVPlays = new PageInfo(tvPlays,3);
        request.setAttribute("pageInfo",pageInfoTVPlays);
    }
    return "search";
}
```

第一个方法 main()是查询所有电影和电视剧，当进入首页后就会自动调用该方法查询出数据库中的数据；第二个方法 search()就是搜索这一个功能点的查询方法，通过前端传入的字段，再采用 pagehelper 插件对数据进行分页操作。

```
@RequestMapping("/login")
public String login(HttpServletRequest request){
    String username=request.getParameter("username");
    String password=request.getParameter("password");
    String role=request.getParameter("role");
    if ("0".equals(role)){
        User user=userService.selectByUsername(username);
        if (user!=null){
            if(password.equals(user.getPassword())){
```

```
                request.getSession().setAttribute("user",user);
            }
        }
        return "redirect:/";
    }else {
        Admin admin=adminService.selectByAdminname(username);
        if (admin!=null){
            if (password.equals(admin.getAdminpassword())){
                request.getSession().setAttribute("admin",admin);
            }
        }
        return "adminIndex";
    }

}

@RequestMapping("/signup")
public String signup(User user,HttpServletRequest request){
    if (userService.insert(user)>0){
        User user1=userService.selectByUsername(user.getUsername());
        request.getSession().setAttribute("user",user1);
    }
    return "redirect:/";
}
```

第一个方法 login()是用户登录，用户在登陆界面输入账号和密码进行登录后端执行逻辑处理；第二个方法 signup()就是用户注册的方法，用户在注册界面输入注册所需要的字段并点击注册，该方法就将注册的用户数据新添到用户表里面。

```
@RequestMapping("/logout")
public String logout(HttpServletRequest request) {
    request.getSession().invalidate();
    return "redirect:/";
}

@RequestMapping("/movielist")
public String ShowList(@RequestParam(required = false,
value="pn",defaultValue="1")Integer pn,
HttpServletRequest request){
    //从第一条开始 每页查询五条数据
    PageHelper.startPage(pn, 5);
    List<Movie> mediaList = movieService.selectAll();
    //将用户信息放入PageInfo对象里
    PageInfo pageInfo = new PageInfo(mediaList,3);
    request.setAttribute("MoviePageInfo", pageInfo);
    return "movielist";
}
```

```
@RequestMapping("/moviegrid")
public String ShowByMoviegrid(@RequestParam(required = false,
value="pn",defaultValue="1")Integer pn,
HttpServletRequest request){
    //从第一条开始 每页查询五条数据
    PageHelper.startPage(pn, 12);
    List<Movie> mediaList = movieService.selectAll();
    //将用户信息放入PageInfo对象里
    PageInfo pageInfo = new PageInfo(mediaList,3);
    request.setAttribute("movieGridPageInfo", pageInfo);
    return "moviegrid";
}

@RequestMapping("/moviegridfw")
public String ShowByMoviegridfw(@RequestParam(required = false,
value="pn",defaultValue="1")Integer pn,
HttpServletRequest request){
    //从第一条开始 每页查询五条数据
    PageHelper.startPage(pn, 12);
    List<Movie> mediaList = movieService.selectAll();
    //将用户信息放入PageInfo对象里
    PageInfo pageInfo = new PageInfo(mediaList,3);
    request.setAttribute("movieGridfwPageInfo", pageInfo);
    return "moviegridfw";
}
```

第一个方法 logout()是用户退出登录，用户点击退出登录时调用的方法；第二个方法 showlist()、showByMoviegridfw()和 showByMoviegrid()就是展示所有电影和电视剧的方法，并通过 pagehelper 插件对其进行分页处理。

```
@RequestMapping("/movieComment")
@ResponseBody
public MovieComment movieComment(@RequestParam String title,
@RequestParam Integer star0, @RequestParam String content,
@RequestParam Integer userid, @RequestParam Integer movieid,
HttpServletResponse response,HttpServletRequest request)
throws IOException {
    Date date=new Date();
    SimpleDateFormat sdf=new SimpleDateFormat("yyyy-MM-dd hh:mm:ss");
    String time=sdf.format(date);
    MovieComment movieComment=new MovieComment(userid,movieid,
    time,star0,title,content);
    movieCommentService.insert(movieComment);
    request.getSession().setAttribute("commentUsername",
```

```
    userService.selectByPrimaryKey(userid).getUsername());
    return movieComment;
}

@RequestMapping("/tvplayComment")
@ResponseBody
public TVPlayComment tvplayComment(@RequestParam String title,
@RequestParam Integer star, @RequestParam String content,
@RequestParam Integer userid, @RequestParam Integer tvid,
HttpServletResponse response,HttpServletRequest request)
throws IOException {
    Date date=new Date();
    SimpleDateFormat sdf=new SimpleDateFormat("yyyy-MM-dd hh:mm:ss")
    String time=sdf.format(date);
    TVPlayComment tvPlayComment=new TVPlayComment(userid,tvid,
    time,star,title,content);
    tvPlayCommentService.insert(tvPlayComment);
    request.getSession().setAttribute("commentUsername",
    userService.selectByPrimaryKey(userid).getUsername());
    return tvPlayComment;
}
```

第一个方法 movieComment()是用户对电影添加评论的页面调用的方法,用户在评论页面输入相应的内容，点击提交后端则会收到数据并添加到相应的数据表中；第二个方法 tvplayComment()是用户对电视剧添加评论的页面调用的方法，与电影类似。

7.2 代码优化分析

刚写代码的时候，发现代码的一个算法的命名让别人看根本看不懂什么意思，于是统一修改算法名字，让别人一眼就可以看出来这个算法是做什么用，同时让算法的名字采用驼峰命名，使得代码更美观。

刚开始的时候，写着写着，发现代码的嵌套层次太多，就自己仔细思考一下，想到一个可以减少嵌套而达到一样的效果，修改之后，很明显代码减少，更方便审阅。

最开始的时候，有很多地方都用公共的代码，一开始没注意到，写着写着就发现，于是就把这段代码封装成一个公共的方法，前端的代码就封装成一个 txt 文本，在前端页面采用<%@include file="head.txt"%>实现代码的复用性。这样既省时间，读者阅读代码也更方便。

还有的时候一些字段根本不需要从前端传过来，在后端可以直接获取到，这个时候就不要从前端传过去，比如，userId 等，在后端取既保证安全，又降低性能等。

这里使用的是 mybatis，大家写 SQL 语句的时候尽量使用#{字段}，不要使用

${字段},因为这样对系统的安全性有着很大的提升作用，${字段}容易被别人侵入，容易被别人动自己的数据库，反之使用#{字段}则不会被侵入！这里也不做具体解释，如果感兴趣的话大家可自行去网上百度。

综上所述，代码优化具体有以下几个部分：算法或类名字需通俗易懂、用最少的代码实现相同的功能、增加代码的复用性、重要的字段不要从前端传过来，需从后台直接取、写 SQL 语句尽量使用#{字段}，不要使用${字段}。

8 测试

8.1 测试方案设计

8.1.1 测试策略

再进行测试之前，测试人员应该思考一下，这个系统可能哪里会出现缺陷。那么，测试的时候，就应该重点注意那些地方。并且应该每个功能点都需要测试，计划测试的内容应该逐一去测试，当出现缺陷的时候，应该及时修复，并且当修复完成之后，重走这个测试流程，当测试过程中没有出现任何 bug 那么这个流程才算完整的。

8.1.2 测试进度安排

表 8-1 测试进度安排表

阶段/工作项	开始时间	结束时间	备注
设计测试方案	2020/2/15	2020/2/15	完成
准备测试环境	2020/2/16	2020/2/16	完成
准备测试资源	2020/2/17	2020/2/17	完成
测试执行,收集结果	2020/2/18	2020/2/20	完成
测试报告	2020/2/21	2020/2/21	完成

8.1.3 测试资源

测试的时候，应该为测试人员搭建好应用的环境和硬软件支持。操作系统是否准备好，测试工具是否准备，相应的测试环境是否配置好，网络情况是否正常，测试时间是否充足，是否有测试人员等一系列的资源安排。

8.1.4 关键测试点

表 8-2 关键测试点表

需求编号	测试需求编号	测试需求描述	优先级
用户登录注册页面			
1	登录	实现影评网站的登录功能	高
2	注册	实现影评网站的注册功能	高
首页			
1	电影和电视剧展示	实现电影和电视剧的显示	高
2	电影排行榜	实现高分电影排行	高
3	搜索	实现电影或电视剧搜索功能	中
电影或电视剧详情页			
1	电影或电视剧收藏	用户收藏电影或电视剧	高
2	电影或电视剧评论	用户评论电影或电视剧	高

续表 8-2

需求编号	测试需求编号	测试需求描述	优先级
电影或电视剧分类页			
1	分页	电影或电视剧实现分页功能	低
用户中心管理页面			
1	查看信息	用户查看个人信息	中
2	查看收藏列表	用户查看个人收藏列表	中
3	查看评论列表	用户查看个人评论列表	中
4	修改信息	用户修改个人信息	中
5	安全管理	用户修改个人登录密码	高
管理页面管理员中心			
1	安全管理	管理员修改个人登录密码	高
2	管理电影列表	管理员管理电影列表	中
3	管理电影评论列表	管理员管理电影评论列表	中
4	管理电视剧评论列表	管理员管理电影评论列表	中
5	管理员列表	管理员查看所有管理员	低
6	添加电影	管理员添加电影	高

8.2 测试用例构建

8.2.1 测试用例编写约定

当测试过程中出现缺陷，需等完成之后进行再次测试，直到相应过程没有缺陷出现。测试的时候必须按照相应的规定测试，不能随意测试。当测试数据库之时可找开发人员帮忙测试。

8.2.2 测试用例设计

每一个测试用例都应该设计好对应的流程，并且说明当时状态，最好还需要测试结果和测试结论等。

8.2.3 关键测试用例

表 8-3 用户登录或注册测试用例

功能说明	用户登录或注册
操作流程	用户进入网站，点击登录或者注册，输入对应的信息，并且点击登录或者注册
初始状态说明	用户登录之后具有评论和收藏、管理个人中心的权限
测试结果	点击注册或登录，成功登录，并且拥有这些权限
测试结论	通过

表 8-4 搜索功能测试用例

功能说明	搜索功能
操作流程	进入首页，选择对应的搜索类型，输出关键字，点击搜索
初始状态说明	无需登录
测试结果	点击搜索，搜索结果成功显示出来
测试结论	通过

表 8-5 电影和电视剧分类测试用例

功能说明	电影和电视剧分类
操作流程	进入首页，点击选择对应的分类类型
初始状态说明	无需登录
测试结果	点击之后，页面成功跳到对应分类页面
测试结论	通过

表 8-6 电影或电视剧收藏测试用例

功能说明	电影或电视剧收藏
操作流程	进入首页，随意点击电影或电视剧进入对应详情页，点击相应按钮
初始状态说明	需要用户登录
测试结果	点击收藏按钮，收藏成功，并在个人中心可以查看得到
测试结论	通过

表 8-7 电影或电视剧评论测试用例

功能说明	电影或电视剧评论
操作流程	进入首页，随意点击电影或电视剧进入对应详情页，点击写评论按钮，并且填写相应信息，点击提交
初始状态说明	需要用户登录
测试结果	提交成功，并且出现了自己提交的评论
测试结论	通过

表 8-8 户安全管理测试用例

功能说明	用户安全管理
操作流程	点击个人中心，点击 profile，输入相应信息并保存
初始状态说明	需要用户登录
测试结果	保存之后退出登录，使用修改后的密码成功登录
测试结论	通过

表 8-9　管理员安全管理测试用例

功能说明	管理员安全管理
操作流程	用户登录到管理员页面，并且进入修改密码页面，输入对应的信息，点击保存
初始状态说明	需要管理员登录
测试结果	保存之后退出登录，使用修改后的密码成功登录
测试结论	通过

表 8-10 测试用例

功能说明	添加电影
操作流程	用户登录到管理员页面，并且进入添加电影页面，输入对应的信息，点击提交
初始状态说明	需要管理员登录
测试结果	成功提交，并且在电影列表里面也可以看到
测试结论	通过

8.2.4 测试用例维护

当在测试过程中遇到问题，应当及时修改测试用例状态，并且及时通知开发人员前来修改。

1. 在同一个项目中，当以后修改或添加新的功能点，就应该新增或修改测试用例；

2. 若后期弃用一些功能点，也应该同时删除测试用例；

3. 当使用期间出现 bug，应新增相应的 bug 测试用例。

9 总结与展望

9.1 设计工作总结

从最开始的选题，当时选题时是比较纠结，因为不知道到底应该选哪类的题目，于是突然看到旁边的人在看电影，于是乎就决定写一个电影评论网站。

然而就迎来需求分析设计部分，当时在网上查阅资料，思考前端如何设计，后端数据库又如何设计，又应该写哪些功能点，花了很久的时间，脑海里也呈现一个基本的构造。电影评论网站，那么最重要的就是影评模块，于是乎就查看各大网站，最后确定，在寻常的评论里面添加一个星级评分的功能，这样就感觉会显得更突出点。

紧接着又到了编码部分，因为在学校就学了 SSM 框架，于是就搭建一个 SSM 框架。刚开始的时候，编码很顺利，知道需要采用模态框的时候，就被卡在这一部分，不知道模态框该怎么设计才好，然后通过询问同学，才知道如何设计，这是编码的时候遇到第一个难题，第二个难题则是在编写评论那一块的时候，知道如何进行添加，但是回显的时候就卡在那里，对一些数据返回之后的拼接还是不熟悉，经过一系列的询问、百度，终于找到了解决方案，于是这个难题也解决。

最后就是编写论文的时候，在这个阶段还是挺顺利，不会的就会去找指导老师询问。

经过这一段时间，成功的把毕业论文写完，从心中出现的第一种心情就是开心，并且在这段过程中学到很多，并且在此过程中拥有很多不同的体验。

在这次毕设中，从最开始的选题到最后的论文编写，这一系列的任务都是自己独立完成，当然，中间遇到一些问题我也会询问同学、老师们。在这次的毕设中，学到了许多，比如：

1、对于前端接收后端报文

2、前端再进行拼接，后端对数据库的多表查询

3、一些事物的处理

4、一些数据如何采用分页显示

5、查询数据表的时候如何会使数据查询更安全。

6、对于一些必传字段都应该进行非空判断

7、前后端的报文尽量使用 json 数据传输

9.2 未来工作展望

在以后的工作中，可能需要能够完善这个项目，比如可以在这个里面添加电影票售票功能，并且可以添加一些权限控制，并且在注册的时候可以采用第三方短信接口。对于后端框架，也可以采用更流行的 spring boot 框架替换，前端页面再进行优化等。前后端代码进行分离。只有经过了这些优化之后，这个项目才

可以拿的出手。

希望在未来能更进一步，并且在工作过程中会仔细阅读开发设计文档。并且会在工作中学到更多东西，毕竟有一句话说的好，往事只有经历，你才会懂得其中的道理。

纸上得来终觉浅，以后的工作中，产生一种欲望，那就是在工作中遇到更多的困难。只有自己遇到困难，并去解决它们，那样，我们才能在工作中学到更多东西。活到老，学到老，这也是我们这一行业的必须遵守的原则，一旦你落后，就会被这个时时进步的社会所淘汰。

未来的工作，但愿能一直在这个行业走下去，并且能一直学习下去，学习是我们这个行业必需做的一件事，往往，只有在工作中才能学到更多东西，才能时时进步。

谢 辞

在这里，我想对以下几位人表示感谢！

第一、指导老师尧艳华与李文奇，在做这个毕设的过程中，有一些不懂的事情，都会去找她帮忙，然后她如果当时没时间，但是她一旦有空回来就会帮我解决问题，并且在写论文的过程中，有几个模块不太理解什么意思，我就找她，她也认认真真地解答这些事。所以我挺感谢她的。

第二、父母，在做毕设的时候，都是早上六点半起床做，因为白天也要上班，父母就一大早起来为我准备早饭什么的，以前不懂事，现在懂事，所以由衷地感谢父母。

第三、高中同学，有一些材料，我找不到，于是她那有相应的材料，她就帮我整理好给我。

最后，我在这里向他们发自有心底地感谢。指导老师尧艳华与李文奇、爸爸妈妈、好朋友谢谢你们！

参考文献

[1] 张宏升.“JavaEE 软件技术”课程的“2+2”创新教学模式研究[J]. 现代信息科技,2017,1(04):78-80.

[2] 唐权. SSM 框架在 JavaEE 教学中的应用与实践[J]. 福建电脑,2017,33(12):93-94+61.

[3] 李杉,贾彦平,达虎. Mybatis 逆向工程在 JavaEE 中的应用[J]. 通讯世界,2017(24):342.

[4] 朱海明. 基于 SSH 框架的 JavaEE 项目代码生成工具的研究与实现[J]. 数字技术与应用,2017(11):57-58+61.

[5] 李宏利,李梅,鱼晓. JavaEE 技术课程资源建设方法研究[J]. 黑龙江教育(高教研究与评估),2018(02):60-61.

[6] 郝平. 基于 JavaEE 的学生社团管理系统的设计与实现[J]. 信息与电脑(理论版),2018(03):103-104+107.

[7] 韦佳佳,任海鹏,孙宇. JavaEE 在轻量级智慧校园架构设计中的应用[J]. 太原学院学报(自然科学版),2018,36(01):50-53.

[8] 陈佳,简岩,杨道平. 基于 JavaEE 的桐城交友网站的设计与实现[J]. 科技经济导刊,2018,26(13):27-28.

[9] Reto Meier. Professional Android 2 Application Development[M]. Birmingham, UK: Wrox, 2010

[10] Hohzaki R, Maehara H. A single-shot game of multi-period inspection[J]. European Journal of Operational Research, 2017, 207(3): 1410-1418

[11] Martin Ngobye Computing Static Slice for Java Programs.2018

[12] Art Tayl or. JDBC Database Programing with J2EE[M].电子工业出版社,2016.

[13] Kathy Sierra, Bert Bates. Absolute Java[M].北京:中国电力出版社.2018.

[14] 疯狂软件. Spring+MyBatis 企业应用实战[M].北京:电子工业出版社,2017.

[15]肖睿,吴振宇. BOOTSTRAP 与 JQUERY UI 框架设计[M].北京:中国水利水电出版社,2017

[16] 刘增辉. MyBatis 从入门到精通[M]. 北京：电子工业出版社，2017.

[17] 胡强,郭标,吴艳文. 触发器在 JavaEE 开发中应用的研究[J]. 合肥师范学院学报,2018,36(03):17-19+26.

[18] 姜桂洪. MySQL 数据库应用与开发[M]. 北京：清华大学出版社，2018.

[19] 张甦. MySQL 王者晋级之路[M]. 北京：电子工业出版社，2018.

[20] 陈晓勇. MySQL DBA 修炼之道[M]. 北京：机械工业出版社，2017.

[21] 贺春旸. MYSQL 管理之道:性能调优、高可用与监控[M]. 北京：机械工业出版社，2016.

[22] 李瑞,徐家喜,卢迪,周涛春. 基于 JavaEE 组合测试系统的设计与实现[J]. 电脑知识与技术,2018,14(14):62-65.

[23] 郭航宇,成丽君. 基于 javaEE 的诗词文化网站系统设计与实现[J]. 电子技术与软件

工程,2018(11):7-8.

[24] 杨黎薇,段洪杰,林国良,崔建文,刘琼仙,邱志刚. 基于 JavaEE 设计模式的烈度速报建设与应用[J]. 世界地震工程,2018,34(02):147-156.

[25] 于博文. 基于计算机软件工程的数据库编程技术[J].中国高新区，2017,（24）：182.

[26] 周文洁. JavaScript 与 jQuery 网页前端开发与设计[M]. 北京：清华大学出版社，2018.

[27] Caliskan . Sevindik.Beginning Spring[M].北京:清华大学出版社,2017.

[28] Dyck.Spring. MVC (A Tutorial series)[M].北京:人民邮电出版社,2017.

[29] 阳波. JavaScript 核心技术开发解密[M]. 北京：电子工业出版社，2018.

[30] 杭建. Java 工程师修炼之道[M]. 北京：电子工业出版社，2018.

附录 A 外文翻译—原文部分

Source from: <<thinking in Java>>

Chapter I Introduction to objects

"The reason why we decompose nature, organize it into various concepts, and classify it according to its meaning is mainly because we are the participants of the agreement that the whole oral communication society abides by. This agreement is fixed in the form of language.... unless we agree with the organization and classification of language information stipulated in this agreement, we can't talk at all." - Benjamin Lee Whorf (1897-1941)

The computer revolution originated from machines, so the generation of programming languages also began from the imitation of machines.

But computers are not just machines. The computer is a tool for brain extension (as Steve jobs often likes to call "the bicycle of the brain"), and it is also a different type of expression medium. As a result, this kind of tool has become more and more like a part of our brain and a form of expression like writing, painting, sculpture, animation, film, etc. Object-oriented programming (0op) is a part of the general trend of taking computer as expression medium.

This chapter introduces you to the basic concepts of OOP, including an overview of development methods. In this chapter, and even in this book, it is assumed that the reader has some programming experience (of course, it must be c). If readers think that they need to do more preparation in programming before reading this book, they should study the multimedia materials of thinking in C which can be downloaded from www.mindview.net.

This chapter introduces the background and supplementary materials. Many people don't feel comfortable doing this kind of programming without knowing the whole picture of object-oriented programming. Therefore, many concepts will be introduced here to help readers have a solid understanding of OOP. However, some people may not understand the whole picture of object-oriented programming until they see the concrete structure. If they do not have the code in hand, they will be in trouble and lose their way. If you belong to the latter group and are eager to get the details of the Java language as soon as possible, you can skip this chapter first, and skip this chapter here, which will not prevent you from writing programs and learning the language. In the end, however, you need to go back to this chapter to supplement what you have learned so that you can understand the importance of objects and how

to use them for design.

1.1 abstract process

All programming languages provide an abstraction mechanism. It can be considered that the complexity of the problem that people can solve directly depends on the type and quality of abstraction. The so-called "type" refers to "what is abstracted?" assembly language is a slight abstraction of the underlying machine. Then many so-called "imperative" languages (such as FORTRAN, basic, C, etc.) are abstractions of assembly language. These languages have been greatly improved on the basis of assembly language, but the main abstractions they make still require to be based on the structure of the computer when solving problems, rather than on the structure of the problems to be solved. The programmer must establish a relationship between the machine model (in the solution space, where you model the problem, such as a computer) and the model of the actual problem to be solved (in the problem space, where the problem exists, such as a business). It is laborious to build such a mapping, and this is not the inherent function of programming language, which makes the program hard to write and expensive to maintain, and also produces the whole "programming method" industry as a by-product.

Another way to model a machine is to model only the problem to be solved. Early programming languages, such as LISP and APL, chose to consider some specific views of the world (corresponding to "all problems are ultimately lists" or "all problems are algorithmic"). Prolog transforms all problems into decision chains. In addition, the language of constraint based programming and the language of graphics symbol operation are also produced (the latter is proved to be too restrictive). These methods are good solutions for the specific types of problems they are trying to solve, but once they go beyond their specific areas, they will not be able to do it.

The object-oriented approach goes one step further by providing programmers with tools to represent elements in the problem space. This representation is so generic that programmers are not limited to any particular type of problem. We call the elements in the problem space and their representation in the solution space "objects". You also need objects that can't be compared to problem space elements. ） The essence of this idea is that a program can apply itself to a specific problem by adding new types of objects. Therefore, when you read the code describing the solution, you are also reading the expression of the problem. This is a more flexible and powerful language abstraction than the language we used before. As a result, OOP allows problems to be described based on the problem, rather than the computer running the solution. But it's

still connected to computers: every object looks a little bit like a microcomputer - it has States and operations, and users can ask objects to perform these operations. If you want to make analogies of objects in the real world, it seems good to say that they all have characteristics and behaviors.

Alan Kay once summed up the five basic features of Smalltalk, the first successful object-oriented language and the language Java is based on. These features represent a pure object-oriented programming approach:

(1) Everything is an object. Treat an object as a strange variable that can store data, and you can also ask it to perform operations on itself. In theory, you can extract any conceptual component (dog, building, service, etc.) of the problem to be solved and represent it as an object in the program.

(2) Programs are collections of objects that tell each other what to do by sending messages. To request - an object, you must send a - message to that object. More specifically, you can think of a message as a call request to a method of a specific object.

(3) Each object has its own storage of other objects. In other words, you can create new types of objects by creating packages that contain existing objects. Therefore, we can build a complex system in the program and hide its complexity behind the simplicity of the object.

(4) Each object has its type. According to the general saying, "every object is an instance of a class", where "class" is synonymous with "type". The most important difference between each class and other classes is "what kind of message can be sent to it".

(5) All objects of a certain type can receive the same message. This is a meaningful statement, you will see later. Because objects of type "circle" are also objects of type "geometry", a "circle" object must be able to accept messages sent to the "geometry" object. This means that you can write code that interacts with "geometry" and automatically processes everything related to geometry properties. This kind of substitutability is one of the most powerful concepts in OOP.

Booch gives a more concise description of objects: objects have state, behavior and identity. This means that each object can have internal data (they give the state of the object) and methods (they generate behavior), and each object can be uniquely distinguished from other objects, specifically, each object has a unique address in memory.

Reuse implementation

It takes a lot of experience and observation to produce a reusable object, but once you have such a design, it can be reused. Code reuse is one of the greatest benefits that object-oriented provides.

The easiest way to reuse a class is to directly use an object of that class, so just put an object of that class into a new class. For example: create a member object. Using existing classes to synthesize new classes is called composition. If a combination is dynamic, it is often referred to as aggregation. A combination is often seen as a has is relationship, such as a car owning an engine.

Combination brings great flexibility. Member objects of new classes are usually declared as private, which makes them inaccessible to client programmers using new classes. Composition, not inheritance, is a priority in development.

inherit

Based on the existing class, copy it, and then create a new class by adding and modifying the copy is called inheritance. When the source class changes, the modified copy (subclass) also reflects these changes.

Type not only describes the constraints acting on an object set, but also the relationship with other types. Two types can have the same characteristics and behavior, but one of them may have more characteristics than the other and can handle more messages. Inheritance uses the concept of base type and subtype to express the similarity between these types. A base type contains properties and behaviors shared by all its subtypes. You can create base types to represent the core concepts of some objects in the system, and subtypes to represent different ways of implementation.

When you inherit an existing type, you create a new type. This new type not only contains all members of the existing type, but also more importantly, it copies the interface of the base class. That is, messages that can be sent to base class objects can also be sent to subclass objects. Because the type of the message sent to the class knows the type of the class, it means that the subclass has the same type as the base class. The behavior of changing the method of the base class is called coverage.

It's the relationship between "one" and "like one"

Subclass object to replace the base class object, called: pure broken substitution. It is often called the principle of substitution. In a sense, this is the ideal way to deal with inheritance. We often call the relationship between the base class and the subclass in this case: is-a is a relationship. To determine whether to inherit or not is to determine whether is-a can be used to describe the relationship between classes.

Some subclasses need to extend the base class, and this new type can also replace the base class, but this substitution is not perfect, because the base class cannot access the new methods of the subclass. This is called an is-liek-a relationship. Subclasses have new methods, so subclasses and base classes are not exactly the same.

When you see the principle of substitution, it's easy to think that pure substitution is the only feasible way, and in fact, it's a good way to design in this way. But you will often find that it is also obvious that you have to add new methods to the interface of subclasses. The usage scenarios of the two methods should be quite obvious.

Interchangeable objects with polymorphism

When dealing with the type hierarchy, we often want to treat an object as the object of its base class instead of its specific type. Enables you to write code that does not depend on a specific type. Such code will not be affected by adding new types, and adding new types is the most common way to extend an object-oriented program to handle new situations once and for all.

However, there are still some problems when subclass objects look at base class objects. The compiler cannot know which code to execute at compile time. This is the key. When sending such a message, the programmer does not want to know which code is executed. If he does not need to know which code is executed, when adding a new type, he can run different code without changing the method calling it.

It is impossible for the compiler to generate function calls in the traditional sense. In OOP, only when the program is running can the address of the code be determined. Therefore, when a message is sent to a generalized object, other mechanisms must be used. This is both late binding. Java uses a small piece of special code instead of an absolute address call. This code uses the information stored in the object to calculate the address of the method body. In this way, according to the content of this code, each object can have different behavior. When a message is sent to an object, the object knows what the message should do.

The process of subclass as its base class is called upward transformation.

附录 B 外文翻译—译文部分

第一章 对象导论

“我们之所以将自然界分解，组织成各种概念，并按其含义分类，主要是因为我们是整个口语交流社会共同遵守的协定的参与者，这个协定以语言的形式固定下来....除非赞成这个协定中规定的有关语言信息的组织和分类，否则我们根本无法交谈。” —Benjamin Lee Whorf (1897 ~ 1941)

计算机革命起源于机器，因此，编程语言的产生也始于对机器的模仿。

但是，计算机并非只是机器那么简单。计算机是头脑延伸的工具(就像 Steve Jobs 常喜欢说的“头脑的自行车”一样)，同时还是一-种不同类型的表达媒体。因此，这种工具看起来已经越来越不像机器，而更像我们头脑的一部分，以及一种如写作、绘画、雕刻、动画、电影等--样的表达形式。面向对象程序设计(Object-oriented Programming, OOP)便是这种以计算机作为表达媒体的大趋势中的组成部分。

本章将向读者介绍包括开发方法概述在内的 OOP 的基本概念。本章，乃至本书中，都假设读者已经具备了某些编程经验(当然不一-定是 C 的)。如果读者认为在阅读本书之前还需要在程序设计方面多做些准备，那么就应该去研读可以从 www.MindView.net 网站上下载的《C 编程思想》(Thinking inC)的多媒体资料。

本章介绍的是背景性的和补充性的材料。许多人在没有了解面向对象程序设计的全貌之前，感觉无法轻松自在地从事此类编程。因此，此处将引入许多概念，以期帮助读者扎实地了解 OOP。然而，还有些人可能在看到具体结构之前，无法了解面向对象程序设计的全貌，这些人如果没有代码在手，就会陷于困境并最终迷失方向。如果你属于后面这个群体，并且渴望尽快获取 Java 语言的细节，那么可以先越过本章一在 此处越过本章并不会妨碍你编写程序和学习语言。但是，你最终还是要回到本章来补充所学知识，这样才能够了解到对象的重要性，以及怎样使用对象进行设计。

1.1 抽象过程

所有编程语言都提供抽象机制。可以认为，人们所能够解决的问题的复杂性直接取决于抽象的类型和质量。所谓的“类型”是指“所抽象的是什么?”汇编语言是对底层机器的轻微抽象。接着出现的许多所谓“命令式”语言(如 FORTRAN、 BASIC, C 等)都是对汇编语言的抽象。这些语言在汇编语言基础上有了大幅的改进，但是它们所作的主要抽象仍要求在解决问题时要基于计算机的结构，而不是基于所要解决的问题的结构来考虑。程序员必须建立起在机器模型(位于“解空间”内，这是你对问题建模的地方，例如计算机)和实际待解问题的模型(位于“问题空间”内，这是问题存在的地方，例如-项业务)之间的关联。建

立这种映射是费力的，而且这不属于编程语言所固有的功能，这使得程序难以编写，并且维护代价高昂，同时也产生了作为副产物的整个“编程方法”行业。

另一种对机器建模的方式就是只针对待解问题建模。早期的编程语言，如 LISP 和 APL,都选择考虑世界的某些特定视图(分别对应于“所有问题最终都是列表”或者“所有问题都是算法形式的”)。PROLOG 则将所有问题都转换成决策链。此外还产生了基于约束条件编程的语言和专门通过对图形符号操作来实现编程的语言(后者被证明限制性过强)。这些方式对于它们所要解决的特定类型的问题都是不错的解决方案，但是一旦超出其特定领域，它们就力不从心了。

面向对象方式通过向程序员提供表示问题空间中的元素的工具而更进了一一步。这种表示方式非常通用，使得程序员不会受限于任何特定类型的问题。我们将问题空间中的元素及其在解空间中的表示称为“对象”。(你还需要-些无法类比为问题空间元素的对象。)这种思想的实质是:程序可以通过添加新类型的对象使自身适用于某个特定问题。因此，当你在阅读描述解决方案的代码的同时，也是在阅读问题的表述。相比以前我们所使用的语言，这是一种更灵活和更强有力的语言抽象。所以，OOP 允许根据问题来描述问题，而不是根据运行解决方案的计算机来描述问题。但是它仍然与计算机有联系:每个对象看起来都有点像--台微型计算机一它具有状态，还具有操作，用户可以要求对象执行这些操作。如果要对现实世界中的对象作类比，那么说它们都具有特性和行为似乎不错。

Alan Kay 曾经总结了第一个成功的面向对象语言、同时也是 Java 所基于的语言之--的 Smalltalk 的五个基本特性，这些特性表现了--种纯粹的面向对象程序设计方式:

(1)万物皆为对象。将对象视为奇特的变量，它可以存储数据，除此之外，你还可以要求它在自身上执行操作。理论上讲，你可以抽取待求解问题的任何概念化构件(狗、建筑物、服务等)，将其表示为程序中的对象。

(2)程序是对象的集合，它们通过发送消息来告知彼此所要做的。要想请求-一个对象，就必须对该对象发送一-条消息。更具体地说，可以把消息想像为对某个特定对象的方法的调用请求。

(3) 每个对象都有自己的由其他对象所构成的存储。换句话说，可以通过创建包含现有对象的包的方式来创建新类型的对象。因此，可以在程序中构建复杂的体系，同时将其复杂性隐藏在对象的简单性背后。

(4) 每个对象都拥有其类型。按照通用的说法，“每个对象都是某个类(class)的-一个实例(instance)”,这里“类”就是“类型”的同义词。每个类最重要的区别于其他类的特性就是“可以发送什么样的消息给它”。

(5)某-特定类型的所有对象都可以接收同样的消息。这是一句意味深长的表述，你在稍后便会看到。因为“圆形”类型的对象同时也是“几何形”类型的对

象，所以一个“圆形”对象必定能够接受发送给“几何形”对象的消息。这意味着可以编写与“几何形”交互并自动处理所有与几何形性质相关的事物的代码。这种可替代性(substitutability) 是 OOP 中最强有力的概念之一。

Booch 对对象提出了一个更加简洁的描述:对象具有状态、行为和标识。这意味着每-个对象都可以拥有内部数据(它们给出了该对象的状态)和方法(它们产生行为)，并且每-一个对象都可以唯一地与其他对象区分开来，具体说来，就是每一一个对象 在内存中都有一个唯-的地址。

复用具体实现。

产生一个可复用的对象需要丰富的经验和观察力的，但是你一旦有了这样的设计，它就可以复用。代码复用时面向对象提供的最了不起的优点之一。

最简单的复用某个类的方式就是直接使用该类的一个对象，因此也而已将那个类的一个对象置于某个新的类中。比如：创建一个成员对象。使用现有的类合成新的类称为：组合（composition）。如果组合是动态发生的，那么它通常称为聚合（aggregation）。组合通常被视为拥有（has-is）关系，比如：汽车拥有引擎。

组合带来极大的灵活性。新类的成员对象通常都被声明为 private，使得使用新类的客户端程序员不能访问他们。开发时优先考虑组合，而不是继承。

继承。

现有的类为基础，复制它，然后通过添加和修改这个副本来创建新类就称为：继承。当源类发生改变时，被修改的副本（子类）也会反映出这些变动。

类型不仅仅与描述了作用于一个对象集合上的约束条件，同时还有与其他类型之间的关系。两个类型可以有相同特性和行为，但是其中一个特性可能比另一个含有更多的特性，而且可以处理更多的消息。继承使用基类型和子类型的概念表示了这种类型之间的相似性。一个基类型包含其所有子类型所共享的特性和行为。可以创建基类型来表示系统中某些对象的核心概念，子类型来表示实现的各种不同方式。

当继承现有的类型时，也就是创造了新的类型。这个新的类型不仅包含现有类型的所有成员，而且更重要的就是它复制了基类的接口。也就是说，所以可以发给基类对象的消息同时也可以发送给子类对象。由于通过发送给类的消息的类型可知类的类型，意味着子类与基类具有相同的类型。改变基类的方法的行为称：覆盖。

是“一个”和“像一个”的关系。

子类对象来替代基类对象，称为：纯碎替代。通常称为替代原则。在某种意义上，这是处理继承的理想方法。我们经常将这种情况下的基类与子类的关系称为：is-a 是一个关系。判断是否继承，就是要确定是否可以使用 is-a 来描述类

之间的关系。

有些子类需要扩展基类，这个新的类型也可以替代基类，但是这种替代并不完美，因为基类无法访问子类的新的方法。这种称为像一个（is-liek-a）关系。子类有新的方法，所以说子类和基类不是完全相同。

当你看到替代原则时，很容易会认为纯碎替代是唯一可行的方式，而且事实上，用这种方式设计是很好的。但是你会时常发现，同样显然的是你必须在子类的接口中添加新方法，两种方法的使用场景应该是相当明显的。

伴随多态的可互换对象。

在处理类型的层次结构时，经常想把一个对象不当做它所属的特定类型来对待，而是将其当做其基类的对象来对待。使得可以写出不依赖于特定类型的代码。这样的代码是不会受添加新类型影响的，而且添加新类型是扩展一个面向对象程序一遍处理新情况的最常用方式。

但是，子类对象看待基类对象时，仍然存在一些问题。编译器在编译时不可能知道执行那一个代码的。这就是关键所在，当发送这样的消息时，程序员并不想知道那一段代码被执行，如果不需要知道那一段代码被执行，那么添加新类型时，不需要改变调用它的方法，它就能运行不同的代码。

编译器不可能产生传统意义上的函数调用，在 OOP 中，程序运行时才能够确定代码的地址。所以，当消息发送到一个泛化的对象时，必须采用其他的机制。这既是后期绑定。Java 使用一小段特殊的代码来替代绝对地址调用。这段代码使用在对象中存储信息来计算方法体的地址。这样，根据这一段代码的内容，每一个对象都可以具有不同的行为表现。当向一个对象发送消息时，该对象就能够知道这条消息应该做些什么。

将子类看做是它的基类的过程称为：向上转型。

附录 C 软件使用说明书

该软件主要是可以用来浏览电影和电视剧详情，并且可以看相应的评分，而且当登录后还可以进行收藏和用户评论等功能。所以说，需要使用该网站，最好是注册一个账号，当然，管理员的账号使用系统分配的，无法注册的，想要更好地使用该网站，最好的就是注册账号，让自己有更好的体验感！

当注册后，可以查看所有电影和电视剧，参考排行榜，可以进行分类查询，可以搜索，具体想干什么就看每个用户的意愿了，当然，你还可以进行收藏电影和电视剧，也可以对电影和电视剧进行评论，抒发自己地看法，也可以在用户中心查看自己的信息，查看自己的收藏列表和评论列表，当然，必不可少的是修改自己地信息修改密码等！

然而，当登录了管理员则是可以添加电影，删除电影，评论，用户等。

附录 D 主要源代码

```
@Controller
public class UserController {
    @Autowired
    private UserService userService;
    @Autowired
    private UserInfoService userInfoService;
    @Autowired
    private MovieService movieService;
    @Autowired
    private TVPlayService tvPlayService;
    @Autowired
    private MovieCommentService movieCommentService;
    @Autowired
    private TVPlayCommentService tvPlayCommentService;
    @Autowired
    private FavoriteService favoriteService;
    @Autowired
    private AdminService adminService;


    @RequestMapping("/")
    public String main(HttpServletRequest request){
        List<Movie> movieList=movieService.selectAll();
        request.getSession().setAttribute("movies",movieList);
        List<TVPlay> tvPlays=tvPlayService.selectAll();
        request.getSession().setAttribute("tvPlays",tvPlays);
        List<Movie> orders=movieService.searchByOrder();
        request.getSession().setAttribute("orders",orders);
        return "index1";
    }

    @RequestMapping("/search")
    public String search(@RequestParam(value = "search") String
search,@RequestParam(required = false,value="pn",defaultValue="1")Integer pn,
```

```
HttpServletRequest request){
        PageHelper.startPage(pn, 5);
        String table=request.getParameter("show_type");
        request.setAttribute("table",table);
        if ("Movie".equals(table)){
            List<Movie> movies=movieService.searchByLike(search);
            PageInfo pageInfoMovies = new PageInfo(movies,3);
            request.setAttribute("pageInfo", pageInfoMovies);
        }else if ("TVPlay".equals(table)){
            List<TVPlay> tvPlays=tvPlayService.searchByLike(search);
            PageInfo pageInfoTVPlays = new PageInfo(tvPlays,3);
            request.setAttribute("pageInfo",pageInfoTVPlays);
        }
        return "search";
    }

    @RequestMapping("/login")
    public String login(HttpServletRequest request){
        String username=request.getParameter("username");
        String password=request.getParameter("password");
        String role=request.getParameter("role");
        if ("0".equals(role)){
            User user=userService.selectByUsername(username);
            if (user!=null){
                if(password.equals(user.getPassword())){
                    request.getSession().setAttribute("user",user);
                }
            }
            return "redirect:/";
        }else {
            Admin admin=adminService.selectByAdminname(username);
            if (admin!=null){
                if (password.equals(admin.getAdminpassword())){
                    request.getSession().setAttribute("admin",admin);
                }
```

```
            }
            return "adminIndex";
        }

    }

    @RequestMapping("/signup")
    public String signup(User user,HttpServletRequest request){
        if (userService.insert(user)>0){
            User user1=userService.selectByUsername(user.getUsername());
            request.getSession().setAttribute("user",user1);
        }
        return "redirect:/";
    }

    @RequestMapping("/logout")
    public String logout(HttpServletRequest request) {
        request.getSession().invalidate();
        return "redirect:/";
    }

    @RequestMapping("/movielist")
    public String ShowList(@RequestParam(required = false,value="pn",defaultValue="1")Integer pn, HttpServletRequest request){
        //从第一条开始 每页查询五条数据
        PageHelper.startPage(pn, 5);
        List<Movie> mediaList = movieService.selectAll();
        //将用户信息放入 PageInfo 对象里
        PageInfo pageInfo = new PageInfo(mediaList,3);
        request.setAttribute("MoviePageInfo", pageInfo);
        return "movielist";
    }

    @RequestMapping("/moviegrid")
    public String ShowByMoviegrid(@RequestParam(required =
```

```
false,value="pn",defaultValue="1")Integer pn, HttpServletRequest request){
        //从第一条开始 每页查询五条数据
        PageHelper.startPage(pn, 12);
        List<Movie> mediaList = movieService.selectAll();
        //将用户信息放入 PageInfo 对象里
        PageInfo pageInfo = new PageInfo(mediaList,3);
        request.setAttribute("movieGridPageInfo", pageInfo);
        return "moviegrid";
    }

    @RequestMapping("/moviegridfw")
    public String ShowByMoviegridfw(@RequestParam(required =
false,value="pn",defaultValue="1")Integer pn, HttpServletRequest request){
        //从第一条开始 每页查询五条数据
        PageHelper.startPage(pn, 12);
        List<Movie> mediaList = movieService.selectAll();
        //将用户信息放入 PageInfo 对象里
        PageInfo pageInfo = new PageInfo(mediaList,3);
        request.setAttribute("movieGridfwPageInfo", pageInfo);
        return "moviegridfw";
    }

    @RequestMapping("/tvplaylist")
    public String Show(@RequestParam(required =
false,value="pn",defaultValue="1")Integer pn, HttpServletRequest request){
        //从第一条开始 每页查询五条数据
        PageHelper.startPage(pn, 5);
        List<TVPlay> mediaList = tvPlayService.selectAll();
        //将用户信息放入 PageInfo 对象里
        PageInfo pageInfo = new PageInfo(mediaList,3);
        request.setAttribute("tvPlayListPageInfo", pageInfo);
        return "tvplaylist";
    }

    @RequestMapping("/moviesingle")
```

```
    public String moviesingle(HttpServletRequest request){
        String movieid1=request.getParameter("movieid");
        Integer movieid=Integer.parseInt(movieid1);
        Movie movie=movieService.selectByPrimaryKey(movieid);
        int ms= (int) Math.round(movie.getMoviestar());
        List<MovieComment>
movieComments=movieCommentService.selectAllByMovieId(movieid);
        List<String> usernames=new ArrayList<>();
        for (MovieComment movieComment:movieComments){

usernames.add(userService.selectByPrimaryKey(movieComment.getUserid()).getUse
rname());
        }
        if (request.getSession().getAttribute("user")!=null){
            List<Favorite> favorites=favoriteService.selectAll();
            for (Favorite favorite:favorites){
                if
(favorite.getMovieid()==movieid&&favorite.getUserid()==((User)request.getSession()
.getAttribute("user")).getUserid()){
                    request.setAttribute("movieStatus",favorite.getStatus());
                }
            }
        }

        request.getSession().setAttribute("usernames",usernames);
        request.getSession().setAttribute("movieComments",movieComments);
        request.getSession().setAttribute("ms",ms);
        request.getSession().setAttribute("movie",movie);
        return "moviesingle";
    }

    @RequestMapping("/seriessingle")
    public String seriessingle(HttpServletRequest request){
        String tvid1=request.getParameter("tvid");
        Integer tvid=Integer.parseInt(tvid1);
```

```
        TVPlay tvPlay=tvPlayService.selectByPrimaryKey(tvid);
        int ts= (int) Math.round(tvPlay.getTvstar());
        List<TVPlayComment>
tvPlayComments=tvPlayCommentService.selectAllByTVId(tvid);
        List<String> usernames=new ArrayList<>();
        for (TVPlayComment tvPlayComment:tvPlayComments){

usernames.add(userService.selectByPrimaryKey(tvPlayComment.getUserid()).getUse
rname());
        }
        if (request.getSession().getAttribute("user")!=null){
            List<Favorite> favorites=favoriteService.selectAll();
            for (Favorite favorite:favorites){
                if
(favorite.getTvplayid()==tvid&&favorite.getUserid()==((User)request.getSession().g
etAttribute("user")).getUserid()){
                    request.setAttribute("tvPlayStatus",favorite.getStatus());
                }
            }
        }


        request.getSession().setAttribute("usernames",usernames);
        request.getSession().setAttribute("tvPlayComments",tvPlayComments);
        request.getSession().setAttribute("ts",ts);
        request.getSession().setAttribute("tvPlay",tvPlay);
        return "seriessingle";
    }

    @RequestMapping("/movieComment")
    @ResponseBody
    public MovieComment movieComment(@RequestParam String title,
@RequestParam Integer star0, @RequestParam String content, @RequestParam
Integer userid, @RequestParam Integer movieid, HttpServletResponse
response,HttpServletRequest request) throws IOException {
```

```
        Date date=new Date();
        SimpleDateFormat sdf=new SimpleDateFormat("yyyy-MM-dd hh:mm:ss");
        String time=sdf.format(date);
        MovieComment movieComment=new MovieComment(userid,movieid,time,star0,title,content);
        movieCommentService.insert(movieComment);

request.getSession().setAttribute("commentUsername",userService.selectByPrimaryKey(userid).getUsername());
        return movieComment;
    }

    @RequestMapping("/tvplayComment")
    @ResponseBody
    public TVPlayComment tvplayComment(@RequestParam String title, @RequestParam Integer star, @RequestParam String content, @RequestParam Integer userid, @RequestParam Integer tvid, HttpServletResponse response,HttpServletRequest request) throws IOException {
        Date date=new Date();
        SimpleDateFormat sdf=new SimpleDateFormat("yyyy-MM-dd hh:mm:ss");
        String time=sdf.format(date);
        TVPlayComment tvPlayComment=new TVPlayComment(userid,tvid,time,star,title,content);
        tvPlayCommentService.insert(tvPlayComment);

request.getSession().setAttribute("commentUsername",userService.selectByPrimaryKey(userid).getUsername());
        return tvPlayComment;
    }

    @RequestMapping("/typeList")
    public String type(@RequestParam(required = false,value="pn",defaultValue="1")Integer pn,@RequestParam String type,HttpServletRequest request){
        //从第一条开始 每页查询五条数据
```

```
        PageHelper.startPage(pn, 5);
        //将用户信息放入 PageInfo 对象里
        List<Movie> movieLists=movieService.searchByTypeLike(type);
        PageInfo pageInfo = new PageInfo(movieLists,3);
        request.setAttribute("typePageInfo", pageInfo);
        request.setAttribute("typeName", type);
        return "type";

    }

    @RequestMapping("/userdetails")
    public String userdetails(@RequestParam Integer userid,HttpServletRequest request){
        UserInfo userInfo=userInfoService.selectByUserId(userid);
        request.getSession().setAttribute("userInfo",userInfo);
        return "userdetails";
    }

    @RequestMapping("/userprofile")
    public String saveUserInfo(@RequestParam Integer userid,HttpServletRequest request){
        UserInfo userInfo=userInfoService.selectByUserId(userid);
        request.getSession().setAttribute("userInfo",userInfo);
        return "userprofile";
    }

    @RequestMapping("/saveUserInfo")
    public String saveUserInfo(HttpServletRequest request,@RequestParam Integer userid,@RequestParam String firstname,@RequestParam String lastname,@RequestParam String address){
        UserInfo userInfo=new UserInfo(userid,firstname,lastname,address);
        userInfoService.insert(userInfo);
        UserInfo userInfo1=userInfoService.selectByUserId(userid);
        request.getSession().setAttribute("userInfo",userInfo1);
        return "userprofile";
```

```
    }

    @RequestMapping("/modifyPassword")
    public String modifyPassword(HttpServletRequest request,@RequestParam Integer userid,@RequestParam String oldPassword,@RequestParam String newPassword){
        User user=userService.selectByPrimaryKey(userid);
        User user1=new User();
        user1.setPassword(newPassword);
        user1.setUserid(userid);
        user1.setUsername(user.getUsername());
        user.setEmail(user.getEmail());
        if (user.getPassword().equals(oldPassword)){
            userService.updateByPrimaryKey(user1);
            return "userprofile";
        }else {
            request.getSession().setAttribute("msg","密码错误！");
            return "userprofile";
        }
    }

    @RequestMapping("/userrate")
    public String userrate(HttpServletRequest request,@RequestParam Integer userid,@RequestParam(required = false,value="pn",defaultValue="1")Integer pn){
        //从第一条开始 每页查询三条数据
        PageHelper.startPage(pn, 3);
        //将用户信息放入 PageInfo 对象里
        List<MovieComment> movieComments1=movieCommentService.selectAllByUserId(userid);
        List<Movie> movies1=new ArrayList<>();
        PageInfo pageInfo = new PageInfo(movieComments1,3);
        for (MovieComment movieComment:movieComments1){

movies1.add(movieService.selectByPrimaryKey(movieComment.getMovieid()));
        }
```

```
        request.setAttribute("movieCommentsPageInfo", pageInfo);
        request.setAttribute("movies1",movies1);
        return "userrate";
    }

    @RequestMapping("/deleteFavoriteByMovieId")
    public Favorite deleteFavoriteByMovieId(@RequestParam Integer
movieid,@RequestParam Integer userid){
        favoriteService.deleteFavoriteByMovieId(movieid,userid);
        return new Favorite();
    }

    @RequestMapping("/deleteFavoriteByTVPlayId")
    public Favorite deleteFavoriteByTVPlayId(@RequestParam Integer
tvplayid,@RequestParam Integer userid){
        favoriteService.deleteFavoriteByTVPlayId(tvplayid,userid);
        return new Favorite();
    }

    @RequestMapping("/addFavoriteMovie")
    public Favorite addFavorite(@RequestParam Integer movieid,@RequestParam
Integer userid){
        Favorite favorite=new Favorite();
        favorite.setMovieid(movieid);
        favorite.setUserid(userid);
        favorite.setStatus(1);
        favoriteService.insert(favorite);
        return favorite;
    }

    @RequestMapping("/addFavoriteTVPlay")
    public Favorite addFavoriteTVPlay(@RequestParam Integer
tvplayid,@RequestParam Integer userid){
        Favorite favorite=new Favorite();
        favorite.setTvplayid(tvplayid);
```

```
            favorite.setUserid(userid);
            favorite.setStatus(1);
            favoriteService.insert(favorite);
            return favorite;
        }

        @RequestMapping("/userfavoritelist")
        public String userfavoritelist(@RequestParam(required = false,value="pn",defaultValue="1")Integer pn,@RequestParam Integer userid,HttpServletRequest request){
            //从第一条开始 每页查询五条数据
            PageHelper.startPage(pn, 2);
            List<Favorite> favorites = favoriteService.selectByUserId(userid);
            List<TVPlay> tvPlays=new ArrayList<>();
            List<Movie> movies=new ArrayList<>();
            //将用户信息放入 PageInfo 对象里
            PageInfo pageInfo = new PageInfo(favorites,3);
            for (Favorite favorite:favorites){
                if (favorite.getMovieid()==0){

tvPlays.add(tvPlayService.selectByPrimaryKey(favorite.getTvplayid()));
                }else {

movies.add(movieService.selectByPrimaryKey(favorite.getMovieid()));
                }
            }
            request.setAttribute("favoriteMovies",movies);
            request.setAttribute("favoriteTVPlays",tvPlays);
            request.setAttribute("userfavoritelist", pageInfo);
           return "userfavoritelist";
        }

        @RequestMapping("/userfavoritegrid")
        public String userfavoritegrid(@RequestParam(required = false,value="pn",defaultValue="1")Integer pn,@RequestParam Integer userid,
```

```
HttpServletRequest request){
        //从第一条开始 每页查询五条数据
        PageHelper.startPage(pn, 5);
        List<Favorite> favorites = favoriteService.selectByUserId(userid);
        List<TVPlay> tvPlays=new ArrayList<>();
        List<Movie> movies=new ArrayList<>();
        //将用户信息放入 PageInfo 对象里
        PageInfo pageInfo = new PageInfo(favorites,3);
        for (Favorite favorite:favorites){
            if (favorite.getMovieid()==0){

tvPlays.add(tvPlayService.selectByPrimaryKey(favorite.getTvplayid()));
            }else {

movies.add(movieService.selectByPrimaryKey(favorite.getMovieid()));
            }
        }
        request.setAttribute("favoriteMovies",movies);
        request.setAttribute("favoriteTVPlays",tvPlays);
        request.setAttribute("userfavoritegrid", pageInfo);
        return "userfavoritegrid";
    }

    @RequestMapping("/adminComment")
    public String adminComment(@RequestParam(required =
false,value="pn",defaultValue="1")Integer pn, HttpServletRequest request){
        //从第一条开始 每页查询五条数据
        PageHelper.startPage(pn, 5);
        List<MovieComment> adminMovieComments =
movieCommentService.selectAll();
        //将用户信息放入 PageInfo 对象里
        PageInfo pageInfo = new PageInfo(adminMovieComments,3);
        request.setAttribute("adminComments", pageInfo);
        return "adminComment";
    }
```

```
    @RequestMapping("/adminTVComment")
    public String adminTVComment(@RequestParam(required = false,value="pn",defaultValue="1")Integer pn, HttpServletRequest request){
        //从第一条开始 每页查询五条数据
        PageHelper.startPage(pn, 5);
        List<TVPlayComment> adminTVPlayComments = tvPlayCommentService.selectAll();
        //将用户信息放入 PageInfo 对象里
        PageInfo pageInfo = new PageInfo(adminTVPlayComments,3);
        request.setAttribute("adminTVComments", pageInfo);
        return "adminTVComment";
    }

    @RequestMapping("/adminUser")
    public String adminUser(@RequestParam(required = false,value="pn",defaultValue="1")Integer pn, HttpServletRequest request){
        //从第一条开始 每页查询五条数据
        PageHelper.startPage(pn, 5);
        List<User> users = userService.selectAll();
        //将用户信息放入 PageInfo 对象里
        PageInfo pageInfo = new PageInfo(users,3);
        request.setAttribute("adminUser", pageInfo);
        return "adminUser";
    }

    @RequestMapping("/adminList")
    public String adminList(@RequestParam(required = false,value="pn",defaultValue="1")Integer pn, HttpServletRequest request){
        //从第一条开始 每页查询五条数据
        PageHelper.startPage(pn, 5);
        List<Admin> admins = adminService.selectAll();
        //将用户信息放入 PageInfo 对象里
        PageInfo pageInfo = new PageInfo(admins,3);
        request.setAttribute("adminList", pageInfo);
```

```
        return "adminList";
    }

    @RequestMapping("/adminMovie")
    public String adminMovie(@RequestParam(required = false,value="pn",defaultValue="1")Integer pn, HttpServletRequest request){
        //从第一条开始 每页查询五条数据
        PageHelper.startPage(pn, 5);
        List<Movie> movies = movieService.selectAll();
        //将用户信息放入 PageInfo 对象里
        PageInfo pageInfo = new PageInfo(movies,3);
        request.setAttribute("adminMovie", pageInfo);
        return "adminMovie";
    }

    @RequestMapping("/adminCommentDelete")
    public String adminCommentDelete(@RequestParam Integer moviecommentid){
            movieCommentService.deleteByPrimaryKey(moviecommentid);
            return "redirect:/adminComment";
        }

    @RequestMapping("/adminTVCommentDelete")
    public String adminTVCommentDelete(@RequestParam Integer typlaycommentid){
        tvPlayCommentService.deleteByPrimaryKey(typlaycommentid);
        return "redirect:/adminTVComment";
    }

    @RequestMapping("/adminAddMovie")
    public String adminAddMovie(@RequestParam("movieuri")MultipartFile file,String moviename,String movietype,String movieoverview,Double moviestar){
        String name=file.getOriginalFilename();
        String movieuri="img/"+name;
        try {
```

```
                file.transferTo(new File("C:\\Users\\刘桂华\\IdeaMavenProjects\\movie\\src\\main\\webapp\\img\\"+name));
            } catch (IOException e) {
                e.printStackTrace();
            }
            Movie movie=new Movie();
            movie.setMoviename(moviename);
            movie.setMovieoverview(movieoverview);
            movie.setMoviestar(moviestar);
            movie.setMovietype(movietype);
            movie.setMovieuri(movieuri);
            movieService.insert(movie);
            return "redirect:/adminMovie";
        }

        @RequestMapping("/adminMovieDelete")
        public String adminMovieDelete(@RequestParam Integer movieid){
            movieService.deleteByPrimaryKey(movieid);
            return "redirect:/adminMovie";
        }

        @RequestMapping("/adminUserDelete")
        public String adminUserDelete(@RequestParam Integer userid){
            userService.deleteByPrimaryKey(userid);
            return "redirect:/adminUser";
        }

        @RequestMapping("/adminModifyPassword")
        public String adminModifyPassword(HttpServletRequest request,@RequestParam(value = "mpass")String mpass,@RequestParam(value = "newpass") String newpass){
            Admin admin=(Admin)request.getSession().getAttribute("admin");
            if (admin.getAdminpassword().equals(mpass)){
                admin.setAdminpassword(newpass);
                adminService.updateByPrimaryKey(admin);
```

```
                return "redirect:/adminList";
            }else {
                return "adminPassword";
            }
        }
}
```