**编号2**

工 程 技 术 学 院

School of Engineering and Technology

**专业方向课程**

**系 部: 工程技术学院**

**专业班级: 计科1603**

**姓 名: 宿孟**

**学 号: 201640885**

**指导教师: 陈秀丽**

**2019年9月制**

# 

基于PHP的电影推荐系统的设计与实现

摘要：据调查显示，随着科学技术不断创新和进步，人们的生活步伐也不断加快，造成大多数人休闲方式转变为在家看看电影或电视，但都多数人都有选择电影上没有明确的目标。因此，提供一个更好的电影推荐类的平台显得更加重要[1]。

此网站旨在为用户提供更好的电影推荐，是基于ThinkPHP框架＋MySQL数据库设计开发。经过对国内外电影推荐网站进行分析，去糙取精，结合技术问题，最后主要参照豆瓣并针对于豆瓣电影的内容太多，界面不够美观，管理员的权限等问题作出了改善，最主要的特点是结合了两种方式向用户进行推荐，对于新用户使用电影类型作为标签为用户进行个性化推荐电影，而对于使用一段时间的用户依据存储的使用该网站的信息向其推荐电影。

本网站除了电影推荐还有影评、影视金曲等主要几个模块功能。增设电影相关的影视金曲这一特色，让用户更加方便的享受电影以及相关资源，同时也是一种变相的电影推荐，相互作用，通过歌曲而对电影感兴趣，或通过电影对歌感兴趣。而且用户在注册登录后还可以对观后的电影发表评论，同时也可以根据影评来选择电影。相关数据资源大多数来源于豆瓣等有名网站，比较可靠。

关键词：Thinkphp；mysql；php；豆瓣电影；电影推荐

# 

目 录

[第一章 绪论 1](#_Toc11763)

[1.1选题目的与背景 1](#_Toc32473)

[1.2 国内外研究现状 1](#_Toc31176)

[1.2.1 推荐系统研究现状 1](#_Toc25799)

[1.2.2 电影推荐系统的研究现状 2](#_Toc25173)

[第二章 开发技术与环境 5](#_Toc30774)

[2.1开发技术 5](#_Toc27750)

[2.2开发工具 6](#_Toc2573)

[2.3 开发环境 7](#_Toc12132)

[2.3.1 硬件环境 7](#_Toc15785)

[2.3.2 软件环境 7](#_Toc27754)

[第三章 需求分析 8](#_Toc5670)

[3.1可行性分析 8](#_Toc28395)

[3.1.1 技术可行性 8](#_Toc15074)

[3.1.2 经济可行性 9](#_Toc2967)

[3.2系统性能需求分析 9](#_Toc5464)

[3.3系统功能需求分析 9](#_Toc30566)

[第四章 系统概要设计 11](#_Toc14971)

[4.1 系统结构设计 11](#_Toc16364)

[4.1.1 系统开发结构设计 11](#_Toc25657)

[4.1.2 系统总体设计 11](#_Toc14183)

[4.2 数据库设计 12](#_Toc9168)

[4.2.1数据库表 12](#_Toc31270)

[4.2.2数据库结构设计 15](#_Toc31775)

[4.2.3数据库实体关系图 20](#_Toc8248)

[第五章 详细设计 22](#_Toc19217)

[5.1整体设计 22](#_Toc1414)

[5.2功能模块设计 23](#_Toc25560)

[5.2.1 前台功能模块 23](#_Toc9154)

[5.2.2 后台功能模块 25](#_Toc1942)

[第六章 系统实现 27](#_Toc5332)

[6.1界面设计 27](#_Toc6299)

[6.1.1前台界面设计 27](#_Toc18861)

[6.1.2后台界面设计 34](#_Toc1138)

[6.2 功能实现 36](#_Toc8356)

[第七章 结论 48](#_Toc831)

[参考文献 49](#_Toc8137)

# 第一章 绪论

## 1.1选题目的与背景

伴随着科学技术地极速发展脚步，大数据时代也随之到来，人们越来越倾向于在网络上寻找自己所需要的新闻、视频娱乐、知识等信息。另一方面由于周末出门交通拥挤，更易产生疲惫感，于是造成了大多数人休闲方式转变为在家看看电影或电视。据调查显示，平时或者周末休息的休闲活动看电影、电视、上网所占的比例有很大的增长。然而internet上的资源信息极速填充网络，人们想要从数以万计的资源中轻松快捷地找到自己所需的信息，困难性也随之增加，即所谓的“信息过载”问题[2]。同时人们在进入网站观看视频电影本身就是在于休闲娱乐放松，没有明确的目标，更希望系统网站能够根据自己的个别的搜索关键词或偏好来个性化推荐相关的资源，而不是所花费成本与获取信息不成正比。针对此种现象，推荐系统横空出世。要想推荐系统成功，必须有优秀精准的推荐算法作支撑，使其足够使整个系统高效运行，推荐算法的好坏评判标准在于给用户推荐结果的质量和推荐系统性能。当前，国内外最常用热门的推荐算法有：基于内容推荐算法、基于关联规则的推荐算法、基于协同过滤推荐算法、混合推荐算法[3]，由于这些推荐算法都存在优点与不足，可以根据需求选择和综合使用[4]。

由此看来，开发一个电影推荐类的网站是很有必要的。豆瓣在这方面已经做得比较好了，但还是有些缺陷，比如分类太多，有点杂，不够简洁大气，当然，这是因为豆瓣是生活类的网站。还有管理权限过大，随意删帖等，所以，基于这些问题再次开发了这个更全面，界面简洁，用户使用学习成本低，同时采用标签法针对不同用户不同情形对用户进行个性化推荐的电影网站。

## 1.2 国内外研究现状

### 1.2.1 推荐系统研究现状

随着21世纪，信息科技大爆发的时代的来临，同时信息网具有纵横交错、传播迅速广泛等特点，因此网络上的出现信息超载问题，且愈演愈烈，亟待解决。推荐系统是解决此类问题而提出来的一种智能商务系统，推荐系统主要是从用户主动发出请求，把所需信息发送给互联网系统模式，改变为由系统主动感知认知用户的信息需求，实时跟踪并及时收集相关重要信息，分析用户的可能需求，同时把用户有兴趣的信息展现出来给用户，这样实现网络系统信息资源服务真正达到主动且对口的目标[5]。推荐技术就是由此衍生出的技术，自出现以来，一直受到热捧，成为此领域关注的焦点[6]。

推荐系统从上世纪90年代出现，迄今为止，快速发展。1995年人工智能协会，卡耐基梅隆大学提出Web Watcher,斯坦福大学推出个性化推荐系统LIRA，掀起推荐技术的研究热潮。1997年，AT&T实验室提出基于协同过滤的推荐算法的推荐系统，推动推荐技术的发展[7]。2005年，G.Adomavicius 和 A.Tuzhilin发表了一篇文章《Toward the Next Generation of Rcommender Systems》[8]，在书中对此领域做了整体论述，并评论与阐述了三大推荐算法，同时对他们的不足做出了说明，并对推荐系统的未来发展方向等问题分别发表看法 [9]。而推荐技术真正成功被应用于现实商业中，是在2003年和2007年Google和Yahoo等公司的用户广告方案的实行，并取到很大的效益。而在国内，推荐服务也紧跟其后。2009年，在北京第一个个性化推荐引擎与推荐系统科研团队组成；2011年，百度世界大会，百度提出的未来internet重要战略规划和发展方向中包括了推荐引擎，以及提出将快速推进给用户推送喜爱的网站和使用频繁的应用软件的计划的实施。

推荐系统根据收集到的用户的使用该网站的浏览足迹进行科学的分析，并推测用户对信息资源的喜爱程度，然后筛选过滤出相关联性最大的信息向用户推荐，是一个从已知信息推测到未知信息的过程，为用户减少搜索操作，替用户缩小选择范围，作出更好的选择。但是个性化推荐也有自身的致命问题[10]，难以突破，如：冷启动问题、特征提取问题、模型过拟合问题和数据稀疏[11]，目前还没有很好的解决此类问题的最优方案。但个性化推荐系统发展趋势并未受很大影响，国外比较受欢迎的推荐系统有Grouplens，亚马逊，Facebook和 Ringo等; 近几年国内，对于推荐系统也是高速发展的阶段，并取得不错的佳绩，如淘宝、京东商城、豆瓣、优酷土豆等。

### 1.2.2 电影推荐系统的研究现状

据相关数据表明，通过网络观看视频的人数在中国已占总人数的五分之一左右[12]，使用网络观看视频的比例攀升至百分之六十五以上，网民观看视频平均每周花费十小时左右；与其他观看电影的形式的相比快速增长，所占比例达到百分之九十五之多[13]。

1997年，MovieLens系统，世界第一个电影推荐系统[14] ，由美国明尼苏达大学GroupLens[15]推行，至今仍被广大研究者作为参考标准。美国影片租赁公司Netflix 筹办的推荐算法大赛[16]，推荐系统被这个比赛极大促进，给电影推荐系统注入了新的力量。发展到现在，在国外，YouTube与Hul在个性化推荐方面大量投入人力与物力。最主流的电影推荐网站还有Jinni、IMDB、Cirticker、Movie1ens 、Nanocrowd、RottenTomatoes [17]。下面分析他们的内容和优点，并由此比较它们之间的差异。

1. Jinni

* 内容：语言，国家，主演、导演，是否基于小说与小说的作者，时长，美国电影协会排名，IMDB评分，摘要以及用户评论；
* 优点：用户及电影分组，使用自身检索系统的海量电影相关的信息把电影分成不同类别同时得出用户偏好以及用户与用户之间的关联性大小。

1. IMDB

* 内容：主演，导演，电影类型，上映日期，影评，用户评分，国家，时长，电影排名，片中相关花絮和穿帮片段，影片音轨和版本；
* 优点：电影推荐使用了多重检索、高级搜索方式实现，其中高级搜索涵盖了对标题进行高级搜索、对名称进行高级搜索，以及协同部分重叠搜索三种搜查方式。

1. Criticker

* 内容：导演编剧，电影类型，内容简介，用户评分，国家等；
* 优点：评分采用兴趣指数对用户喜好进行判定，就是系统会为每个用户对已评分的电影建立个人的等级库，系统对用户所有评分电影自动设置等级。

1. MovieLens

* 内容：平均得分，用户评分，电影类型，语言，主演、导演，影评人数以及标签；
* 优点：采用三种搜索方式：分别是对标题、类型搜索和高级搜索，第三个是针对前两者的不足进行优化完善；并且仅仅要求用户给一些电影做出评分，该系统就会根据用户之前的数据主动替用户对其余的电影给出较为准确的估测得分。

1. Nanocrowd

* 内容：演员，导演，类型，评论（包括内外部用户的评论），小说作者，美国电影协会排名，标签，制片商；
* 优点：与众不同的标签体系和定位沙盒。通过每部电影的标签框中的不同标签向用户推荐相关电影，还有小型问卷。定位沙盒利用地图展现了电影或标签之间关联度。

1. Rotten Tomatoes

* 内容：上映日期，时长，简介，票房，演员，导演，类型，排名，作者，评论；
* 优点：答案简单，只有好与坏之分；通过三个群体给每部电影评分而作为推荐的标准：由一般用户、权威影评人和有名影评人评比的电影，参考价值都很大，用户可根据需求来选择参考群体。

在国内，作为已是行业领先的电影推荐网站豆瓣电影，会向注册用户推荐与你兴趣相关度较高的电影；如果用户浏览具体影片，系统会向浏览该影片的用户推荐其他浏览该电影的用户还喜欢的电影；也推出豆列等功能，在豆列里，也有相关推荐。最重要的是豆瓣使用了更好的标签数据，通过标签分类，让电影推荐中同时推荐热门电影被得以解决。标签可以帮助人们更好地描绘事物以及对信息进行分类，便于检索和分享，使向用户推荐的电影更加匹配度更大。除此之外，也有其他较使用量较大的电影推荐系统，如：优酷、爱奇艺和乐视的视频推荐模块。

优酷，虽然使用其推荐算法产生的推荐结果和用户已观看的记录具有较高相关性[18]，但也会把用户已观看过的视频或影片再次推荐给用户，算法在数据过滤方面没有做到严格要求，存在漏洞，用户刷新推荐内容时经常夹杂有完全相同的内容。

乐视，虽也有视频推荐功能，但它并没有对该技术功能给以足够关注度，仅仅作为用户个人中心的一小部分，所占模块十分小，用户很难去关注

爱奇艺，使用三种方式结合完成强大的推荐功能，主要是依据收集的用户的观影偏好与观影数据等数据建模，然后通过模型匹配用户偏好得出结果然后向用户推荐。首先是播放区域下方有相关联的视频，同时在视频结束跳转向用户推荐推荐大量用户在看过此视频后，很大比例的感兴趣的视频。其次是在爱奇艺社区奇谈模块收集用户在奇谈上浏览足迹，推测出用户可能也喜欢的视频，然后进行个性化推荐。最后是为用户建立个人的相关模型，然后得出结论再推荐视频。

# 

# 第二章 开发技术与环境

## 2.1开发技术

1）PHP

PHP是一门Web[脚本语言](http://baike.baidu.com/item/%E8%84%9A%E6%9C%AC%E8%AF%AD%E8%A8%80" \t "_blank),主要被用来作为[Web](http://baike.baidu.com/item/Web" \t "_blank)开发工具。据大量数据表明它是一门易于学习，入门门槛低的编程语言，因此使用也十分[广泛](http://baike.baidu.com/item/%E5%B9%BF%E6%B3%9B" \t "_blank)。PHP 特性如下：

快速执行动态网页：将代码放在HTML文档中形成动态网页并执行，与其他方式比效率更高；

1.跨平台性强：就现有的的数据库和操作系统而言，PHP普遍都支持；

2.可扩展性强：可使用C和C++进行扩展；

3.开源：免费，代码完全开源，开源社区活跃；

4.快捷性：开发网站所需时间短，学习快速人们;

由于PHP有着这几大特性，能够为用户提供强大的功能，因此在网站建设中成为广大编程人员的首选。

2）MySQL数据库

MySQL，一个由MySQLAB公司开发的轻量的[关系型数据库管理系统](http://baike.baidu.com/item/%E5%85%B3%E7%B3%BB%E5%9E%8B%E6%95%B0%E6%8D%AE%E5%BA%93%E7%AE%A1%E7%90%86%E7%B3%BB%E7%BB%9F" \t "_blank)，由于其开源，同时集执行快、成本低、性能高、使用方便、多用户同时操作、跨平台性等特性和优点于一身。

MySQL服务器主要是用来处理大型数据库，与其他数据库服务器相比，它将数据分散于几张表中，因此速度更快，也更加灵活，在高标准严要求的应用场景也是如鱼得水。目前MySQL服务器的功能更趋于全面完善多样。因为安全性良好、运行快等特点，互联网中得到大量使用。所以本网站采用MySQL关系型数据库来存储数据。

3）Apache服务器

Apache是全球使用量最大的操作Web[服务器](http://baike.baidu.com/item/%E6%9C%8D%E5%8A%A1%E5%99%A8" \t "_blank)的软件。在普遍的操作系统平台上都能运行良好，[跨操作系统](http://baike.baidu.com/item/%E8%B7%A8%E5%B9%B3%E5%8F%B0" \t "_blank)与安全方面一直处于领先地位。

Apache具有轻量、运行效率高、性能良好且稳定等优点，一般用作[代理服务器](http://baike.baidu.com/item/%E4%BB%A3%E7%90%86%E6%9C%8D%E5%8A%A1%E5%99%A8" \t "_blank)，进行虚拟域名设置。Apache集成了大量的功能，比如采用[SSL](http://baike.baidu.com/item/SSL" \t "_blank)技术和设置多虚拟站点。Apache的基本单位是[进程](http://baike.baidu.com/item/%E8%BF%9B%E7%A8%8B" \t "_blank)，但由于与[线程](http://baike.baidu.com/item/%E7%BA%BF%E7%A8%8B" \t "_blank)相比较，对系统容量要求更高，效果并不好，但可采取扩充节点的方式加以解决。Apache在服务器市场覆盖率已过半。

4）NodeJs

Node.js是以Javascript作为运行环境，并整合封装了Google V8引擎。V8引擎对于执行JS速度比之前的其他引擎快上数倍，性能良好。Node.js是一个服务器程序，支持多并发，在网络平台上得到广泛使用，效果显著。Node.js轻量化、运行高效，广泛应用在大量数据请求的实时的分布式网络应用。

NodeJS出现以后，使得JS不再仅仅在浏览器客服端使用，更加开拓了在服务器领域的应用场景，因此实现前后端语言统一，操作简单，对习惯了JS语法的程序员来说，编程更加轻松。本系统主要用NodeJS来爬取数据，并存入数据库；采用NodeJS爬取数据非常便利快速，NodeJS拥有许多相关的插件包，安装也十分迅速简单，通过向服务器发送请求，控制并发量，及时获取响应，解析数据都快速方便。

## 2.2开发工具

1）ThinkPHP框架

ThinkPHP是为了给中小型企业简化开发流程，从而得以快速高效的开发web级应用的目标而开发的，最终也达到了该目的。thinkPHP同时还以许多外国的比较好的框架和模式作为参考，从而整合出这套性能高、轻量、简单的框架。ThinkPHP坚持认为简单最好，达到了代码简洁、简单易用的要求。它还增加许多功能，由于其开源免费，使用者囊括中外，群体庞大，社区功能完善。

虽然PHP框架有很多，比如：laravel,Yii等，但thinkphp采用 MVC模式开发，简单易用，可扩展性好，组件插件支持，版本更新快。此平台采用的版本是3.2.3，虽然现在已更新到think PHP5.0.5，但之前比较熟悉thinkPHP3.2.3版本，并且觉得更稳定，所以采用此版本。

2）phpStudy

phpStudy是一个调试PHP环境的程序集成包。把[Apache](http://baike.baidu.com/item/Apache" \t "_blank)、[PHP](http://baike.baidu.com/item/PHP" \t "_blank)、MySQL等程序软件结合在一个包中,功能完善，版本不断更新完善，不同版本包含不同的软件，使用在不同的平台，使用时下载安装便可，它的服务器是FTP，不需要经过服务器复杂的配置过程即可开启，真正做到傻瓜式操作。作为一个 PHP调试环境，方便了PHP的开发，提高开发效率；在用户支持上，可支持多用户。对于开发新手来说，简单方便高效，运行快速，同时可快速配置多站点虚拟域名。最新版的PHPstudy还支持自己定义PHP版本，同时支持多平台。由此PHPStudy使PHP开发更加快捷简单高效，让PHP更加受开发者的青睐。

3）Navicat for MySQL

Navicat for MySQL是为了实现MySQL数据库的数据和操作流程可视化而产生的工具，查看操作更加简单直观，给操作sql语言命令不熟练的程序员带来便利，通过它能很大程度地提高开发效率，缩短开发时间。

Navicat for MySQL能同时连接MySQL、MariaDB数据库，更好地用于这两者的管理与操作。当使用查询语句时采用可视化的直接的SQL编辑工具，不再需要对SQL语法和命令进行强记和复杂的编辑。同时为用户提供云服务，用户能随时随地实时访问，更大限度提高工作效率。Navicat有两种连接方式，分别是可靠的 SSH连接，在多个数据库上实现远程连接共享和直接使用HTTP作为连接通道，与internet服务商连接，然后从中获取服务。

## 2.3 开发环境

### 2.3.1 硬件环境

windows 10操作系统的PC

### 2.3.2 软件环境

1. 操作系统: Windows 10;
2. 浏览器: chrome 57.0.2987.133;
3. 服务器相关工具: phpstudy 集成服务器;
4. 数据库: MySQL 数据库;
5. 前端调试工具：Firebug和chrome开发者工具;
6. 开发语言及插件：PHP、JavaScript、jQuery、BootStrap、Ajax等;
7. IDE： sublime text3编辑器;

# 第三章 需求分析

需求分析是处在在软件系统开发之前、功能需求确定的最后阶段，其主要是为系统确定工作任务，为系统实现提出最终的需求方案。需求分析得出的分析报告作为系统开发的重要指南，指导系统开发的全过程，当然在开发中也有根据实际情况来进行更改，需求分析决定了软件开发的成败与否和最终产品是否能投入使用，达到预期目标。因此，必须使用一些行之有效的措施来对应用程序的需求分析实行更加严格的检查与验证。本章主要从可行性分析、系统性能需求和功能需求进行分析。

## 3.1可行性分析

### 3.1.1 技术可行性

本系统是电影推荐网站，基于PHP框架ThinkPHP3.2.3和MySQL数据库。 PHP是在脚本中嵌入HTML文档，然后运行的服务端语言，大多应用作为 Web编程，可在不同的操作系统中使用。PHP已是所有编程语言中受欢迎程度很高的唯一的脚本语言。据不完全统计在互联网应用网站中，采用PHP作为主要技术超过百分之六十； 近年来，PHP使用人数所占比例远远大于JAVA和.NET,由此可见其已是相当成熟并被大众接受的语言。而以其为技术核心开发的thinkPHP框架，在技术也是十分可靠稳定。

MySQL是小型关系型数据库，同时也是使用多线程、并发量高的数据库系统[19]，而且是具有商用价值、普遍性的数据库。它具有以下功能优点：

1.处理大量记录的大型数据；

2.支持常见的SQL语句，对查询算法优化，查询速率得到大幅度提高；

3.可移植性高，跨平台，支持多种操作系统，轻量级；

4.运行效率高，丰富信息的网络支持；

5.调试、管理，优化简单，易用性强；

6.支持多种开发语言，并为其提供了API；

相对于其他大型数据库，MySQL简单、高效、可靠、易用，同时满足中小型网站开发的需求，能够快速处理上万条数据，据第三方测评，在系统性能也是比其他数据库更好。

因此，对于开发此小型系统，使用PHP+MySQL在技术方面可以保证系统稳定良好运行。

### 3.1.2 经济可行性

此系统主要应用已成熟且开源的技术，无须其他技术相关的费用，且开发快速高效周期短，节约开发成本；使用MVC模式开发，M（数据层）、V（视图层）、C（控制层）分离，后期维护容易、可扩展性强；其次，本系统开发人员较少，后期维护无需太多人员，节约人力财力；因此在后期维护使用和再次迭代开发也不会产生其他较高的费用。

## 3.2系统性能需求分析

本系统是推荐类网站，主要为用户提供热门、最新的电影推荐以及相关推荐服务，所以系统需要能够承受较大的访问量，同时能够较快的响应速度，由于使用PHP的think PHP3.2.3框架，成熟稳定，同时结合MySQL数据库，数据操作性能强大，能够快速处理大量数据。采用Ajax异步刷新技术，完成页面部分刷新，快速响应用户请求，并返回结果。因此能够满足系统的性能需求。

## 3.3系统功能需求分析

作为电影推荐系统，本系统主要是为用户进行个性化推荐电影。为实现此目的，首先，对于新用户如果未注册登录或者已注册登录未完善个人资料，则向其推荐高分、最新或最热门的电影;若已完善资料则向其推荐完善资料所选的喜欢的电影类型相同同时高分的电影；若是老用户则根据用户使用网站的信息向用户推荐电影。

本系统主要功能包括电影管理、推荐排行、用户管理、影视金曲、影评、用户留言评论、注册登录、个人中心。本系统分为前台功能和后台功能，具体如下：

1）前台功能：注册登录、个人中心完善个人资料、电影推荐、搜索、影视金曲、影评、用户评论。前台功能需求图如下：



**图3-1 前台功能需求图**

2）后台功能：主要是管理员对用户管理、菜单管理、电影管理、影视金曲管理、影评管理评论管理等模块进行的添加、更新、删除、批量删除等操作。后台功能需求图如下：

**图3-2 后台功能需求图**

第四章 系统概要设计

概要设计是紧接需求分析完成后面，主要是把需求分析抽象概念转化为具体实体设计的过程，即将系统需求分析所提来的逻辑转化为可实施的方案。系统概要设计主要为下一阶段的编码提供依据，要达到经济、可靠、灵活、系统的要求。本章节将从数据库设计、系统结构设计分析该系统的概要设计。

## 4.1 系统结构设计

### 4.1.1 系统开发结构设计

本系统采用现在比较流行的快速开发中小型网站的thinkphp框架，它是采用单一入口模式、MVC开发模式。MVC模式机制如下图：



**图4-1 MVC模式图**

### 4.1.2 系统总体设计

1. 搜索相关资源：用户打开网站，可以浏览和搜索最新、最热的电影，并浏览电影详情；
2. 浏览使用相关资源：用户可以浏览电影影评和听影视相关的歌曲等；
3. 用户评论：如果用户想要对电影作出评论，首先应该注册登录，然后才能留言评论；
4. 注册登录：用户进入本网站后可注册登录，还可以到个人中心完善自己的资料，以便拥有其他使用权限；
5. 推荐资源：系统根据相关逻辑给用户推荐匹配的电影等资源；
6. 管理员操作：管理员可以进入系统后台，对相关数据实行操作管理：添加、删除、修改、批量删除、排序等。

总体功能框架如下图所示：



**图4-2 总体功能结构框图**

## 4.2 数据库设计

### 4.2.1数据库表

表4-1 用户表tb\_user

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 类型 | 长度 | 空 | 说明 |
| user\_id | int | 8 | 否 | 用户表的主键 |
| username | varchar | 255 | 否 | 用户名 |
| password | varchar | 255 | 否 | 用户密码 |
| email | varchar | 255 | 是 | 用户邮箱 |
| phone | varvhar | 255 | 是 | 用户手机号 |
| lastLoginTime | datetime |  | 否 | 最近登陆时间 |
| status | tinyint | 2 | 否 | 用户状态，是否登陆 |
| pic | varchar | 255 | 否 |  |
| sex | tinyint | 2 | 否 |  |

表4-2 管理员表tb\_admin

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 类型 | 长度 | 空 | 说明 |
| admin\_id | mediumint | 8 | 否 | 管理员表的主键 |
| username | varchar | 255 | 否 | 用户名 |
| password | varchar | 255 | 否 | 用户密码 |
| email | varchar | 255 | 是 | 用户邮箱 |
| phone | varvhar | 255 | 是 | 用户手机号 |
| lastLoginTime | datetime |  | 否 | 最近登陆时间 |
| status | tinyint | 2 | 否 | 用户状态，是否登陆 |

表4-3 电影表tb\_movie

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 类型 | 长度 | 空 | 说明 |
| movie\_id | int | 12 | 否 | 电影表的主键 |
| movie\_name | varchar | 255 | 否 | 电影名 |
| movie\_type | int | 3 | 否 | 电影类型 |
| grade | float | 2 | 是 | 电影评分 |
| pic | varvhar | 255 | 是 | 电影封面图 |
| up\_time | datetime |  | 否 | 上映时间 |
| status | tinyint | 2 | 否 | 电影状态（是否存在） |
| listorder | smallint | 5 | 否 | 序号 |
| big\_pic | varcahr | 255 | 是 | 电影剧照大图 |

表4-4 电影详情表tb\_movie\_detail

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 类型 | 长度 | 空 | 说明 |
| movie\_detail\_id | int | 12 | 否 | 电影详情表的主键 |
| movie\_id | int | 12 | 否 | 电影表主键 |
| actors | varchar | 255 | 否 | 主演 |
| director | varchar | 255 | 否 | 导演 |
| describle | mediumtext |  | 否 | 简介 |
| rank | int | 5 | 否 | 排名 |
| count | int | 50 | 否 | 票房 |

表4-5 影评表tb\_comment

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 类型 | 长度 | 空 | 说明 |
| id | int | 8 | 否 | 影评表的主键 |
| movie\_name | varchar | 255 | 否 | 电影名 |
| avator | varchar | 255 | 否 | 影评人头像 |
| author | varchar | 20 | 否 | 影评人名称 |
| pic | varchar | 255 | 否 | 电影封面图 |
| big\_pic | varchar | 255 | 是 | 电影剧照图 |
| content | varchar | 20 | 否 | 影评文章内容 |
| source | varcahr | 20 | 否 | 影评来源 |
| title | varchar | 100 | 否 | 文章标题 |
| listorder | int | 6 | 否 | 序号 |
| status | tinyint | 2 | 否 | 状态（是否存在） |
| time | tinyint | 1 | 否 | 发表时间 |

表4-6 歌曲表tb\_music

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 类型 | 长度 | 空 | 说明 |
| id | int | 8 | 否 | 歌曲表的主键 |
| music\_name | varchar | 255 | 否 | 歌曲名 |
| singer | varchar | 255 | 否 | 歌手 |
| movie\_name | varchar | 255 | 否 | 电影名 |
| pic | varcahr | 255 | 否 | 歌曲封面图 |
| url | varchar | 255 | 否 | 播放链接 |
| content | mediumtext |  | 是 | 歌词 |
| time\_long | varcahr | 255 | 否 | 歌曲时长 |
| listorder | int | 8 | 否 | 序号 |
| status | tinyint | 2 | 否 | 状态（是否存在） |

表4-7 电影推荐表tb\_rank\_movie

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 类型 | 长度 | 空 | 说明 |
| id | int | 8 | 否 | 电影推荐表的主键 |
| movie\_id | int | 12 | 否 | 电影名 |
| url | varchar | 255 | 是 | 详情链接 |
| movie\_name | varchar | 255 | 否 | 电影名 |
| pic | varcahr | 255 | 是 | 电影封面图 |
| grade | float | (2,1) | 否 | 电影评分 |
| push\_time | datetime |  | 否 | 推荐时间 |
| rank | int | 5 | 否 | 推荐排名 |
| listorder | int | 8 | 否 | 序号 |
| status | tinyint | 2 | 否 | 状态（是否存在） |

表4-8 菜单表tb\_menu

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 类型 | 长度 | 空 | 说明 |
| id | int | 8 | 否 | 电影推荐表的主键 |
| movie\_id | int | 12 | 否 | 电影名 |
| url | varchar | 255 | 是 | 详情链接 |
| movie\_name | varchar | 255 | 否 | 电影名 |
| pic | varcahr | 255 | 是 | 电影封面图 |
| grade | float | (2,1) | 否 | 电影评分 |
| push\_time | datetime |  | 否 | 推荐时间 |
| rank | int | 5 | 否 | 推荐排名 |
| listorder | int | 8 | 否 | 序号 |
| status | tinyint | 2 | 否 | 状态（是否存在） |

表4-9 用户评论表tb\_review

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 类型 | 长度 | 空 | 说明 |
| id | int | 8 | 否 | 电影推荐表的主键 |
| movie\_id | int | 12 | 否 | 电影名 |
| url | varchar | 255 | 是 | 详情链接 |
| movie\_name | varchar | 255 | 否 | 电影名 |
| pic | varcahr | 255 | 是 | 电影封面图 |
| grade | float | (2,1) | 否 | 电影评分 |
| push\_time | datetime |  | 否 | 推荐时间 |
| rank | int | 5 | 否 | 推荐排名 |
| listorder | int | 8 | 否 | 序号 |
| status | tinyint | 2 | 否 | 状态（是否存在） |

### 4.2.2数据库结构设计

-- ----------------------------

-- Table structure for tb\_user

-- ----------------------------

DROP TABLE IF EXISTS `tb\_user`;

CREATE TABLE `tb\_user` (

`user\_id` int(8) unsigned NOT NULL AUTO\_INCREMENT,

`username` varchar(255) NOT NULL,

`password` varchar(255) NOT NULL,

`lastLoginTime` datetime NOT NULL,

`email` varchar(255) DEFAULT NULL,

`phone` varchar(255) DEFAULT NULL,

`status` tinyint(2) NOT NULL,

`pic` varchar(255) DEFAULT NULL,

`sex` tinyint(2) NOT NULL DEFAULT '0',

PRIMARY KEY (`user\_id`)

) ENGINE=MyISAM AUTO\_INCREMENT=15 DEFAULT CHARSET=utf8;

-- ----------------------------

-- Table structure for tb\_admin

-- ----------------------------

DROP TABLE IF EXISTS `tb\_admin`;

CREATE TABLE `tb\_admin` (

`admin\_id` mediumint(8) unsigned NOT NULL AUTO\_INCREMENT,

`username` varchar(20) NOT NULL DEFAULT '',

`password` varchar(32) NOT NULL DEFAULT '',

`lastloginip` varchar(15) DEFAULT '0',

`lastlogintime` datetime DEFAULT '0000-00-00 00:00:00',

`email` varchar(40) DEFAULT '',

`realname` varchar(50) NOT NULL DEFAULT '',

`status` tinyint(1) NOT NULL DEFAULT '1',

PRIMARY KEY (`admin\_id`),

KEY `username` (`username`)

) ENGINE=MyISAM AUTO\_INCREMENT=9 DEFAULT CHARSET=utf8;

-- ----------------------------

-- Table structure for tb\_comment

-- ----------------------------

DROP TABLE IF EXISTS `tb\_comment`;

CREATE TABLE `tb\_comment` (

`id` int(8) unsigned NOT NULL AUTO\_INCREMENT,

`movie\_name` varchar(50) NOT NULL,

`author` varchar(225) DEFAULT NULL,

`content` mediumtext,

`status` tinyint(2) NOT NULL DEFAULT '1',

`time` varchar(255) DEFAULT NULL,

`source` varchar(255) DEFAULT NULL,

`listorder` int(8) NOT NULL,

`title` varchar(255) NOT NULL,

`pic` varchar(255) NOT NULL,

`avator` varchar(255) DEFAULT NULL,

`big\_pic` varchar(255) DEFAULT NULL,

PRIMARY KEY (`id`)

) ENGINE=MyISAM AUTO\_INCREMENT=717 DEFAULT CHARSET=utf8;

-- ----------------------------

-- Table structure for tb\_menu

-- ----------------------------

DROP TABLE IF EXISTS `tb\_menu`;

CREATE TABLE `tb\_menu` (

`menu\_id` int(6) unsigned NOT NULL AUTO\_INCREMENT,

`name` varchar(40) NOT NULL DEFAULT '',

`m` varchar(20) NOT NULL DEFAULT '',

`c` varchar(20) NOT NULL DEFAULT '',

`f` varchar(20) NOT NULL DEFAULT '',

`data` varchar(100) NOT NULL DEFAULT '',

`listorder` int(6) unsigned NOT NULL DEFAULT '0',

`status` tinyint(1) NOT NULL DEFAULT '1',

`type` tinyint(1) unsigned NOT NULL DEFAULT '0',

PRIMARY KEY (`menu\_id`),

) ENGINE=MyISAM AUTO\_INCREMENT=28 DEFAULT CHARSET=utf8;

-- ----------------------------

-- Table structure for tb\_movie

-- ----------------------------

DROP TABLE IF EXISTS `tb\_movie`;

CREATE TABLE `tb\_movie` (

`movie\_id` int(12) unsigned NOT NULL,

`movie\_name` varchar(255) NOT NULL,

`movie\_type` int(3) NOT NULL,

`grade` float(8,1) NOT NULL,

`pic` varchar(255) NOT NULL,

`up\_time` date DEFAULT NULL,

`status` tinyint(2) NOT NULL DEFAULT '1',

`listorder` smallint(5) DEFAULT '0',

`big\_pic` varchar(255) DEFAULT NULL,

PRIMARY KEY (`movie\_id`),

UNIQUE KEY `movie\_name` (`movie\_name`)

) ENGINE=MyISAM AUTO\_INCREMENT=26903993 DEFAULT CHARSET=utf8;

-- ----------------------------

-- Table structure for tb\_movie\_detail

-- ----------------------------

DROP TABLE IF EXISTS `tb\_movie\_detail`;

CREATE TABLE `tb\_movie\_detail` (

`movie\_detail\_id` int(12) unsigned NOT NULL AUTO\_INCREMENT,

`movie\_id` int(12) unsigned NOT NULL,

`actors` varchar(255) NOT NULL,

`director` varchar(255) NOT NULL,

`describle` mediumtext NOT NULL,

`rank` int(20) NOT NULL,

`count` int(50) NOT NULL,

`up\_time` datetime DEFAULT NULL,

PRIMARY KEY (`movie\_detail\_id`)

) ENGINE=MyISAM AUTO\_INCREMENT=255 DEFAULT CHARSET=utf8;

-- ----------------------------

-- Table structure for tb\_music

-- ----------------------------

DROP TABLE IF EXISTS `tb\_music`;

CREATE TABLE `tb\_music` (

`id` int(8) unsigned NOT NULL AUTO\_INCREMENT,

`music\_name` varchar(255) NOT NULL,

`singer` varchar(255) NOT NULL,

`movie\_name` varchar(255) NOT NULL,

`pic` varchar(255) DEFAULT NULL,

`url` varchar(255) DEFAULT NULL,

`content` mediumtext,

`time\_long` varchar(255) NOT NULL,

`listorder` int(8) NOT NULL DEFAULT '0',

`status` tinyint(1) NOT NULL DEFAULT '1',

PRIMARY KEY (`id`)

) ENGINE=MyISAM AUTO\_INCREMENT=4453 DEFAULT CHARSET=utf8;

-- ----------------------------

-- Table structure for tb\_rank\_movie

-- ----------------------------

DROP TABLE IF EXISTS `tb\_rank\_movie`;

CREATE TABLE `tb\_rank\_movie` (

`id` int(8) unsigned NOT NULL AUTO\_INCREMENT,

`movie\_id` int(12) NOT NULL,

`movie\_name` varchar(255) NOT NULL,

`url` varchar(255) DEFAULT NULL,

`pic` varchar(255) DEFAULT NULL,

`grade` double(2,1) NOT NULL,

`listorder` int(8) NOT NULL,

`status` tinyint(2) NOT NULL DEFAULT '1',

`push\_time` datetime NOT NULL,

`rank` int(5) NOT NULL,

PRIMARY KEY (`id`)

) ENGINE=MyISAM AUTO\_INCREMENT=200 DEFAULT CHARSET=utf8;

-- ----------------------------

-- Table structure for tb\_review

-- ----------------------------

DROP TABLE IF EXISTS `tb\_review`;

CREATE TABLE `tb\_review` (

`id` int(8) unsigned NOT NULL AUTO\_INCREMENT,

`movie\_id` int(8) NOT NULL,

`user\_id` int(8) NOT NULL,

`avator` varchar(255) DEFAULT NULL,

`content` varchar(255) NOT NULL,

`time` datetime NOT NULL,

`username` varchar(255) NOT NULL,

PRIMARY KEY (`id`)

) ENGINE=MyISAM AUTO\_INCREMENT=71 DEFAULT CHARSET=utf8;

### 4.2.3数据库实体关系图



**图 4-12 数据库实体关系图**

# 第五章 详细设计

## 5.1整体设计

整体设计是以该系统的整体操作流程来进行设计，包括前台用户和后台管理员对本系统的操作使用。整体详细设计流程图如下：



**图5-1 系统流程图**

## 5.2功能模块设计

### 5.2.1 前台功能模块



**图5-2 前台功能模块图**

1. 注册登录模块

用户进入到该网站后，可以点击头部右侧的注册按钮，进行注册，输入用户名和密码并进行校验，校验成功后，进入登陆页面，用户登录并进行校验，成功后进入网站首页；



**图5-3 注册登录流程图**

1. 个人中心模块

用户注册登录后，可以点击头像或用户名，在个人中心中，完善个人资料，上传个人头像等；

1. 电影模块

搜索电影名称，如果数据库中有概数据，则显示搜索查找到的电影；如果没有，则显示未找到相关电影资源；找到该电影后，可点击进入电影详情页，显示电影详细信息；

1. 电影推荐模块

对于初始使用该网站的用户，若已完善信息中的所喜欢的电影类型使用标签法进行匹配，若未则推荐高分最热的电影；对于老用户，使用协同过滤算法所写的slop one中的思想。

1. 用户评论模块

在电影详情页下方，如果用户已注册并登录，可以对该电影进行评论；如果没有注册登陆，会提醒用户未注册登录；评论后可以在留言评论区看到自己的评论；



**图5-4用户评论流程图**

1. 影视金曲模块

主要是相关的歌曲。点击导航栏菜单中的“影视金曲”，则可进入相关影视的相关歌曲，包括主题曲、插曲、片尾，同时展示电影名、歌手等信息；点击歌曲可进入详情页，可以听该歌曲，并展示歌词；

1. 影评模块

主要是观影人或专业影评人对电影的影评文章。点击导航栏菜单中的“影评”，进入影评列表页，再点击进入影评文章页，阅读相关电影影评；

### 5.2.2 后台功能模块



**图5-5 后台功能模块图**

1. 电影管理模块

主要是管理员对电影模块进行管理，分别是增删改查、批量删除、排序、电影推荐、预览等操作。点击后台左侧菜单“电影管理”，显示数据库中的电影；然后可进行相关操作。具体操作：在上部的搜索框 中搜索电影模糊匹配，将从数据库中搜索匹配条件的数据进行展示，并设置分页机制，若数据条数超过设置的数目，则分页展示；点击每条电影列表的操作中的删除图标，对该条数据进行删除；点击编辑图标，跳转到编辑页面，对该电影进行修改；勾选电影进行推荐，将该电影数据添加到电影推荐排行数据库表中；在每条电影前面的输入框中，输入数字，点击“排序”按钮，电影列表会按从大到小进行排序。以下相同功能不赘述；

1. 菜单管理模块

主要是管理员对前后台菜单进行管理，分别是增删改查、批量删除、排序等操作。点击后台左侧菜单“菜单管理”，显示数据库中的所有的菜单；菜单包括前台菜单和后台菜单；前台菜单是前台页面中的菜单导航，后台菜单是后台中的左侧的各个菜单模块；

1. 相关歌曲管理模块

主要是管理员对歌曲进行管理，分别是增删改查、批量删除、排序、歌曲预览等操作。点击后台左侧菜单“影视金曲管理”，右侧显示所有歌曲，然后对数据进行相关操作；

1. 影评管理模块

主要是管理员对影评进行管理，分别是增删改查、批量删除、排序、影评预览、影评推荐等操作。点击后台左侧菜单“影评管理”，右侧显示所有影评列表信息；

1. 用户管理

主要是管理员对用户进行管理，分别是查询、删除、批量删除、排序等操作。点击后台左侧菜单“用户管理”，右侧显示所有已注册的用户信息，管理员对用户只有查看和删除权限，无法进行编辑更改；

# 第六章 系统实现

## 6.1界面设计

### 6.1.1前台界面设计

1. 首页

由头部、主体组成，头部部分包括菜单导航、注册登录、个人中心；主体部分包括上半部分的最新推荐轮播和下半部分的电影“精彩推荐”与右侧的“电影推荐排行”、“影评推荐”；具体设计如下图所示：



**图6-1 首页界面图**

1. 电影模块

1．电影列表页：由网站公共头部和主体部分组成。主体部分包括顶部的搜索框和下方的“最新推荐”、“最热推荐”的电影列表以及底部的分页部分。具体设计如下图所示：



**图6-2 电影列表界面图**

2.电影详情页：由公共头部部分和主体部分组成。主体部分包括：左侧的电影名称、导演、主演、类型、豆瓣排名、票房、上映时间、电影简介等具体信息以及电影的用户观影之后的评论信息、评论发表区域和右侧侧边栏的相关电影推荐与影评推荐。具体设计如下图所示：



**图6-3 电影详情界面图**

1. 影视金曲模块

1.影视金曲列表页：由公共头部部分和主体部分组成。主体部分包括：左侧侧边栏的文字信息和右侧的主要内容歌曲列表表单；歌曲列表包括歌曲名称、歌手、所属电视剧或电影、时长等信息。具体设计如下图所示：



**图6-4 影视金曲列表界面图**

2.影视金曲详情页：由公共头部部分和影视金曲的主体部分组成。主体部分包括：左侧侧边栏文字描述信息和右侧主要内容中的歌曲封面、歌词和下部分的歌曲播放控制区域。具体详细设计如下图所示：



**图6-5 影视金曲详情界面图**

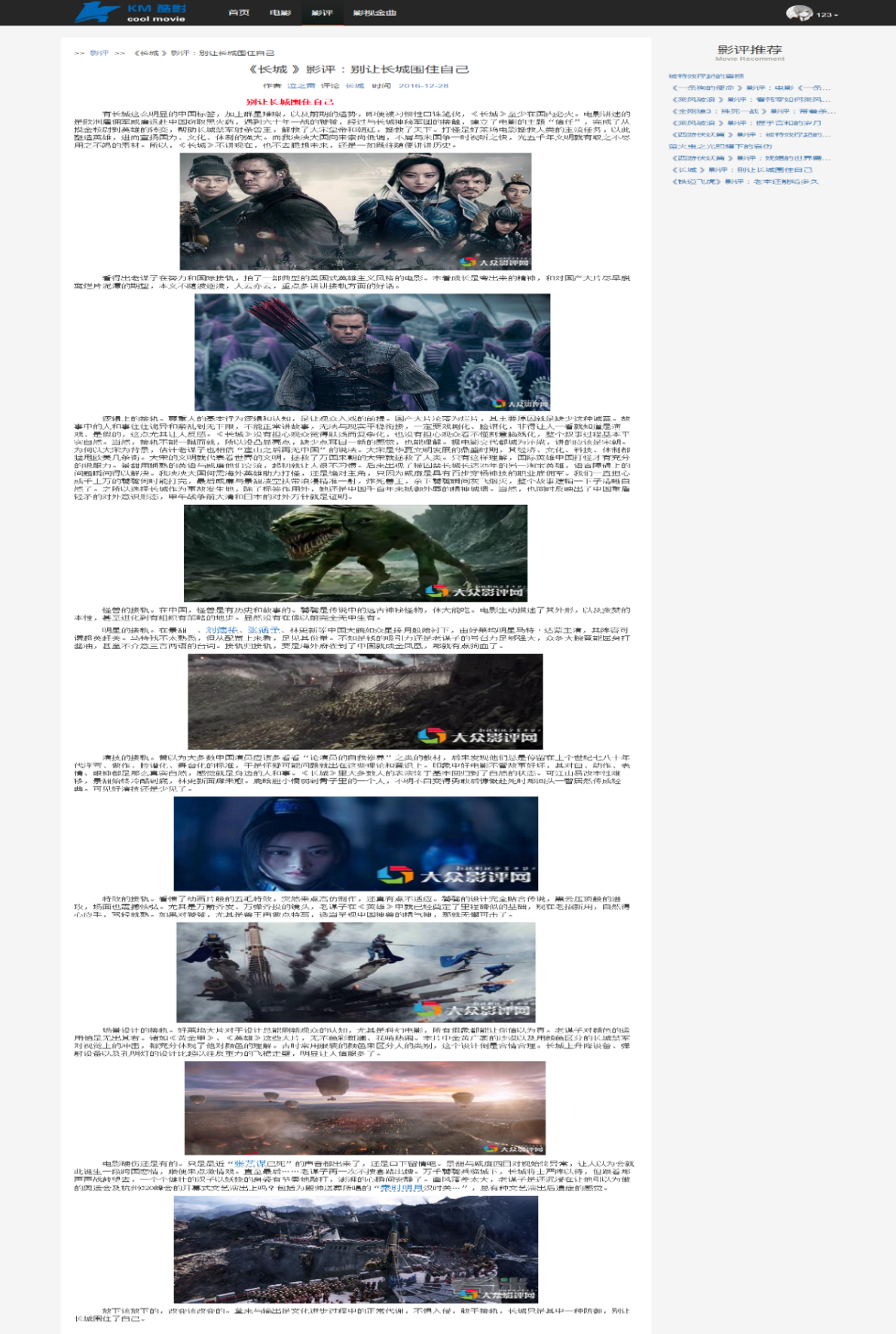
1. 影评模块

1.影评列表页：由公共头部部分和影评的主体部分组成。主体部分包括：左侧主要部分中最热影评推荐、最新影评列表和右侧侧边栏电影推荐排行。具体详细设计如下图所示：



**图6-6 影评列表界面图**

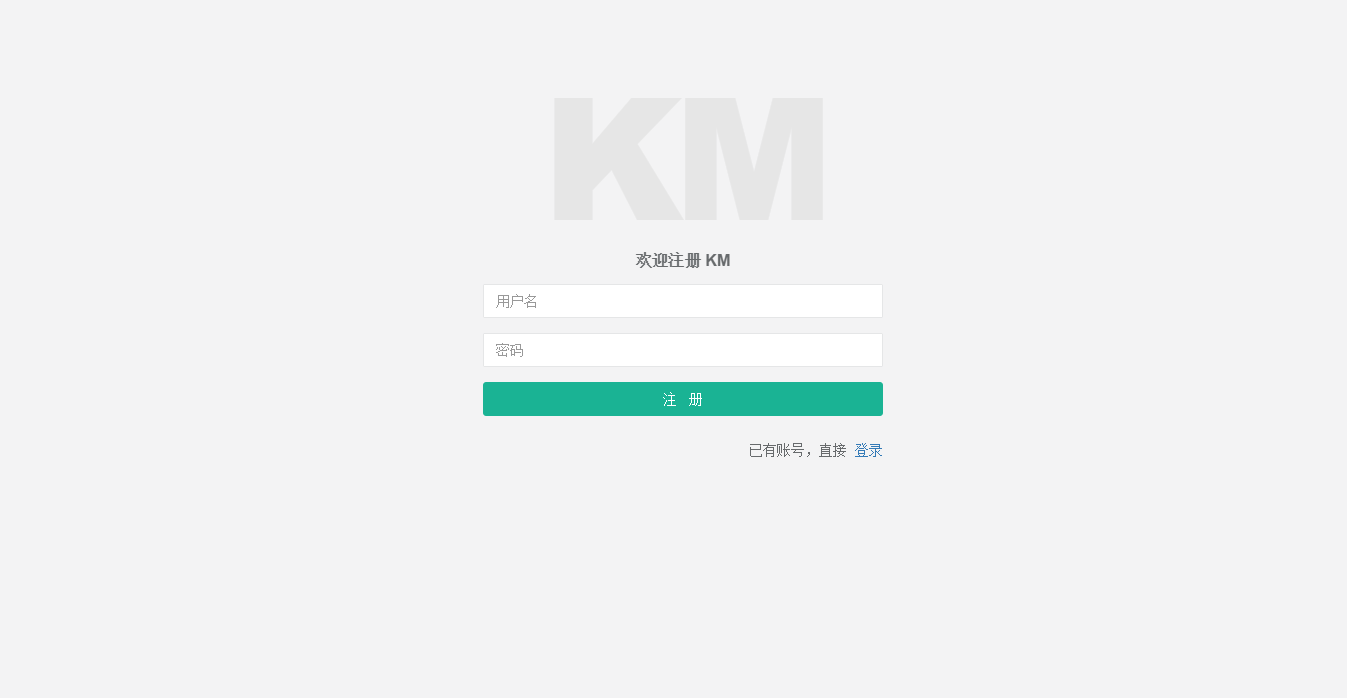
2.影评文章页：由公共头部部分和影评详情页的主体部分组成。主体部分包括：左侧主要内容影评文章和右侧侧边栏影评推荐。具体详细设计如下图所示：



**图6-7 影评详情界面图**

1. 用户模块

1.注册页面：包括用户名和密码输入框，注册按钮和登录链接；



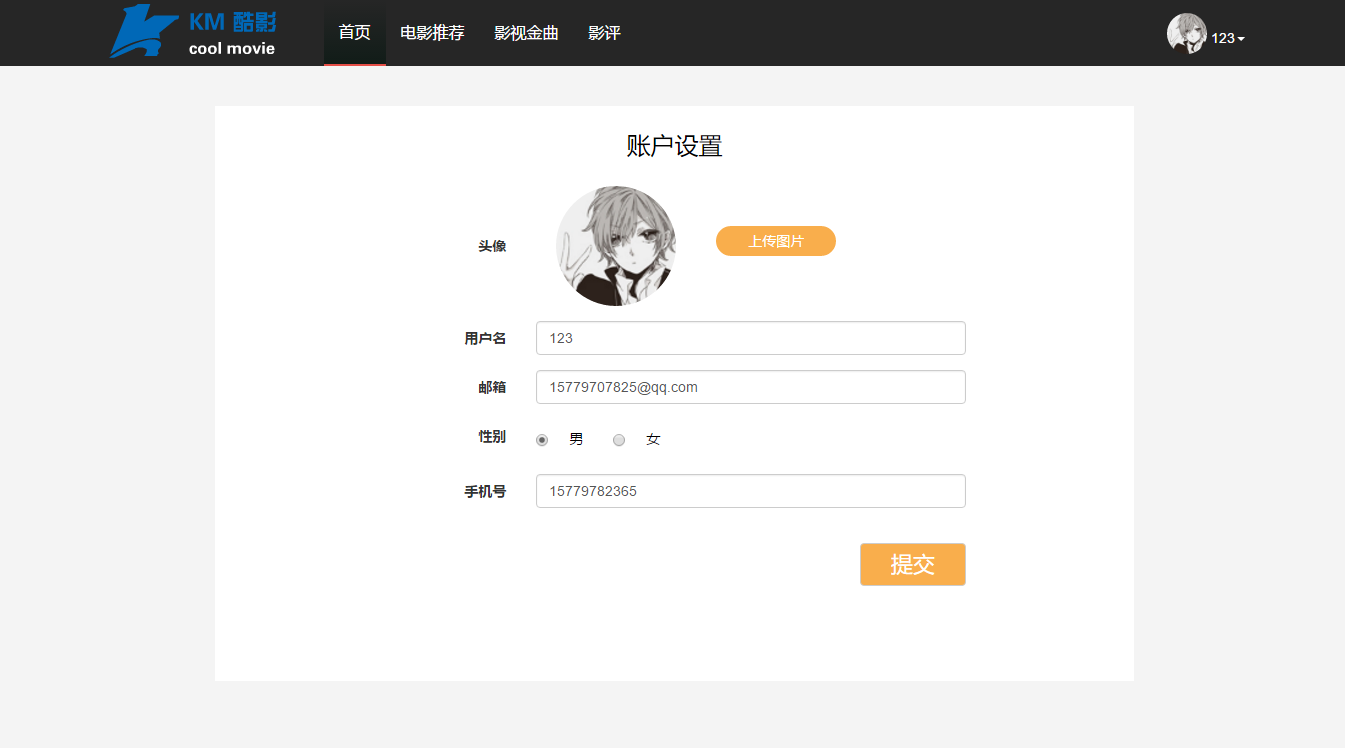
**图6-8用户注册界面图**

2.登录页面：包括用户名和密码输入框，登录按钮和注册链接；



**图6-9 用户登录界面图**

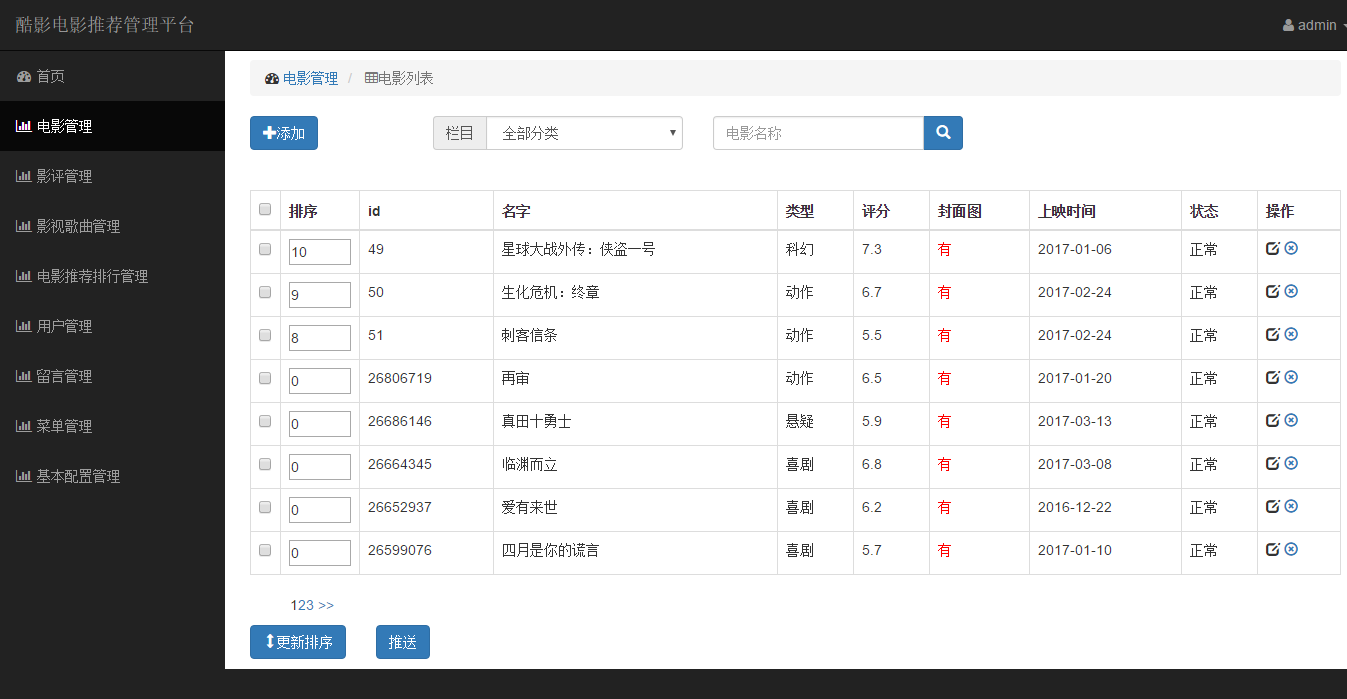
3.个人中心页面：包括头像上传按钮、用户名、邮箱、性别、手机号输入框以及提交按钮；



**图6-10 用户个人中心界面图**

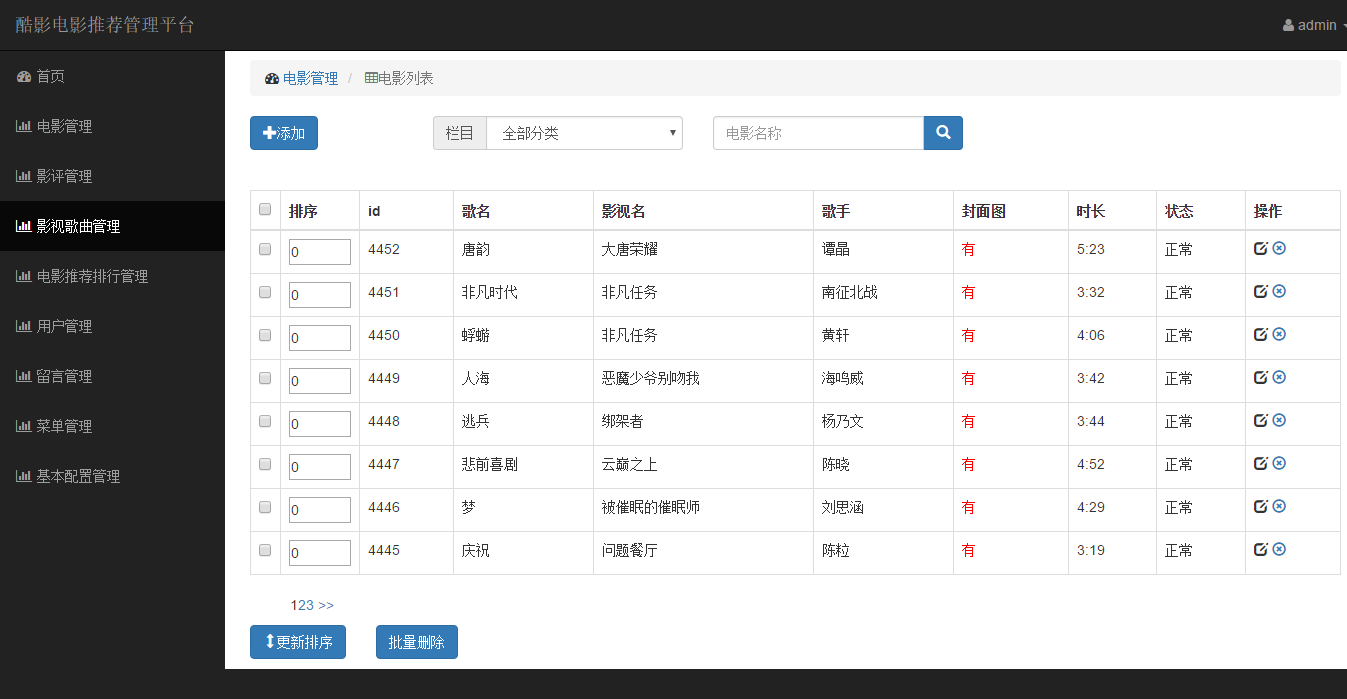
### 6.1.2后台界面设计

1. 电影管理模块：包括电影模块添加、删除、修改、推送、更新排序、搜索、分页等功能操作。



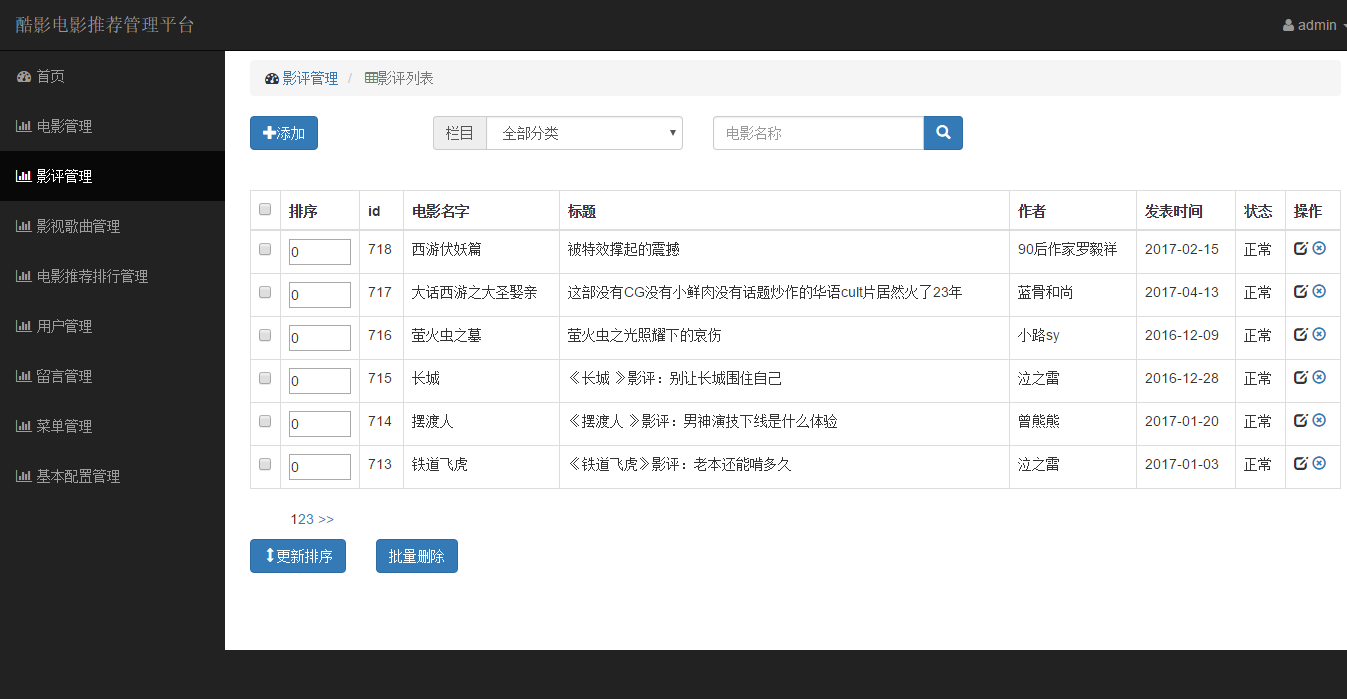
**图6-11 后台电影管理界面图**

1. 影视歌曲管理模块：包括影视歌曲模块添加、删除、修改、推送、更新排序、搜索、分页等功能操作。



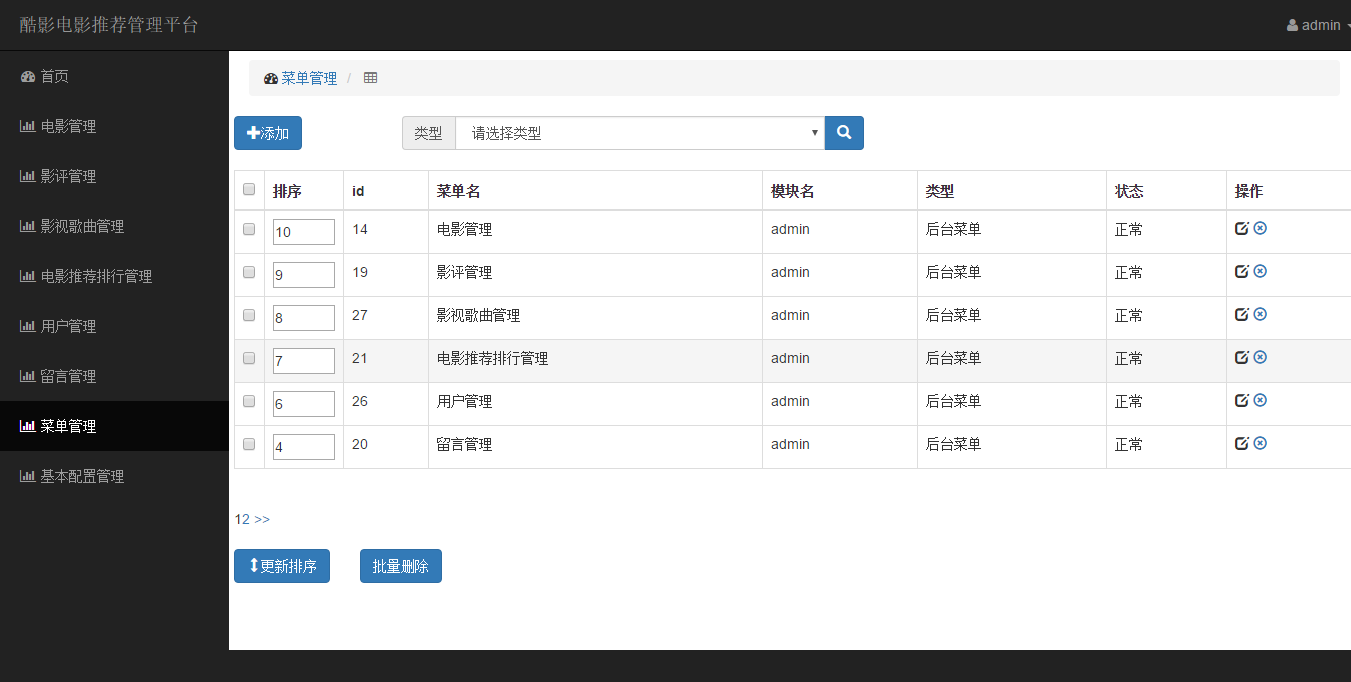
**图6-12 后台影视歌曲管理界面图**

1. 影评管理模块：包括影评模块添加、删除、修改、批量删除、更新排序、搜索、分页等功能操作。



**图6-13 后台影评模块界面图**

1. 菜单模块：包括ID、菜单名、模块、类型等列表项的显示、添加、删除、修改。



**图6-14 后台菜单模块界面图**

## 6.2 功能实现

1. 控制器公共函数与方法

class CommonController extends Controller {

public function \_\_construct() {

parent::\_\_construct();

$this->\_init();

}

private function \_init() {

// 如果已经登录

$isLogin = $this->isLogin();

if(!$isLogin) {

// 跳转到登录页面

$this->redirect('/admin.php?c=login');

}

}

/\*\*

\* 获取登录用户信息

\* @return array

\*/

public function getLoginUser() {

return session("adminUser");

}

/\*\*

\* 判定是否登录

\* @return boolean

\*/

public function isLogin() {

$user = $this->getLoginUser();

if($user && is\_array($user)) {

return true;

}

return false;

}

1. 后台增删改查等相关操作逻辑代码

1．后台首页先管逻辑代码，主要包含分页，数据通过模版渲染到页面，具体见图6-10。

//电影首页后台逻辑

public function index() {

／\*分页逻辑 \*／

$conds['status'] = array('neq',-1);

//地址栏请求第几页，默认为第一页

$page = $\_REQUEST['p'] ? $\_REQUEST['p'] : 1;

$pageSize = 8; //页数

//获取符合查询条件的（电影）数据项

$Movie = D("Movie")->getMovieList($conds,$page,$pageSize);

//获取符合查询条件的（电影）总数

$count = D("Movie")->getMovieCount($conds);

//调用think PHP自身框架封装的分页函数

$res = new \Think\Page($count,$pageSize);

//模版渲染

$pageres = $res->show();

$this->assign('pageres',$pageres);

$this->assign('Movie',$Movie);

//菜单模版渲染

$this->assign('webSiteMenu',D("Menu")->getBarMenus());

$this->display();

}

2.添加模块逻辑代码，主要是点击列表页（首页）中的“添加”按钮进入添加编辑页面，添加编辑表单中的各项信息，完成后在提交，通过以下逻辑代码添加到数据库中，最后前台页面会在数据库中取出并渲染。

public function add(){//电影添加，其他模块类似，不赘述

if($\_POST) {获取POST过来的数据

／\*判断相关数据是否存在，并改出提示\*／

if(!isset($\_POST['movie\_name']) || !$\_POST['movie\_name']) {

return show(0,'电影名不存在');

}

if(!isset($\_POST['movie\_type']) || !$\_POST['movie\_type']) {

return show(0,'电影类型不存在');

}

if(!isset($\_POST['grade']) || !$\_POST['grade']) {

return show(0,'评分不存在');

}

if(!isset($\_POST['pic']) || !$\_POST['pic']) {

return show(0,'缩略图不存在');

}

if(!isset($\_POST['up\_time']) || !$\_POST['up\_time']) {

return show(0,'上映时间不存在');

}

// 更新数据库中具体ID的信息

if($\_POST['movie\_id']) {

return $this->save($\_POST);

}

//通过POST获取的数据插入到数据库中

//此处是插入到movie的主表tb\_movie

$movieId = D("Movie")->insert($\_POST);

if($movieId) {

//若插入成功，再把详细电影信息插入详细表中

//其他模块没有此逻辑操作

$movieContentData['describle'] = $\_POST['describle'];

$movieContentData['movie\_id'] = $movieId;

$movieContentData['count'] = $\_POST['count'];

$movieContentData['rank'] = $\_POST['rank'];

$movieContentData['director'] = $\_POST['director'];

$movieContentData['actors'] = $\_POST['actors'];

$cId = D("MovieDetail")->insert($movieContentData);

if($cId){

return show(1,'新增成功');

}else{

return show(1,'主表插入成功，副表插入失败');

}

}else{

return show(0,'新增失败');

}

}else { //获取电影类型，其他模块没有此逻辑操作

$movie\_type= C("MOVIE\_TYPE");

$this->assign('movie\_type', $movie\_type);

$this->display();

}

}

3.修改更新模块逻辑代码，主要是点击列表页各项的“编辑”图标进入编辑页面显示该列表项的信息，然后修改需要更更改的信息后提交更新，更新数据库数据。

//电影信息更新，其他模块类似，不赘述

//此函数是通过ID获取信息显示在表单中

public function edit() {

$movieId = $\_GET['id'];//获取地址栏中带过来的movie\_id

if(!$movieId) {

// 若不存在直接回到列表页（首页）执行跳转

$this->redirect('/admin.php?c=movie');

}

$movie = D("Movie")->find($movieId);

if(!$movie) {

// 若不存在直接回到列表页（首页）执行跳转

$this->redirect('/admin.php?c=movie');

}

//通过地址栏传过来的ID查找返回结果

$movieDetail = D("MovieDetail")->find($movieId);

if($movieDetail) {

//查找到电影该ID的详细信息，并赋值

$movie['describle'] = $movieDetail['describle'];

$movie['rank'] = $movieDetail['rank'];

$movie['count'] = $movieDetail['count'];

$movie['actors'] = $movieDetail['actors'];

$movie['director'] = $movieDetail['director'];

}

$webSiteMenu = D("Menu")->getBarMenus();

$this->assign('webSiteMenu', $webSiteMenu);

//模版渲染，把具体信息显示到修改编辑表单中

$this->assign('movieType', C("MOVIE\_TYPE"));

$this->assign('movie',$movie);

$this->display();

}

//保存数据，其他模块类似，不赘述

//此函数是获取表单中的值更新到数据库

public function save($data) {

$MovieId = $data['movie\_id'];

unset($data['movie\_id']);

try {

$id = D("Movie")->updateById($MovieId, $data);

$MovieDetailData['describle'] = $data['describle'];

$MovieDetailData['rank'] = $data['rank'];

$MovieDetailData['count'] = $data['count'];

$MovieDetailData['director'] = $data['director'];

$MovieDetailData['actors'] = $data['actors'];

$condId = D("MovieDetail")->updateMvDetailById($MovieId, $MovieDetailData);

if($id === false || $condId === false) {

return show(0, '更新失败');

}

return show(1, '更新成功');

}catch(Exception $e) {

return show(0, $e->getMessage());

}

}

4.更新状态逻辑代码，主要是修改某一列表项的状态，数据库通过状态status值来判断该条数据是否存在，如不存在则在后台页面中不会显示，删除某一项后，该项的状态就为-1或0，表示已删除。

//电影更新状态，其他模块类似，不赘述

public function setStatus() {

try {

if ($\_POST) {

$id = $\_POST['id'];

$status = $\_POST['status'];

if (!$id) {

return show(0, 'ID不存在');

}

$res = D("Movie")->updateStatusById($id, $status);

if ($res) {

return show(1, '操作成功');

} else {

return show(0, '操作失败');

}

}

return show(0, '没有提交的内容');

}catch(Exception $e) {

return show(0, $e->getMessage());

}

}

5.更新状态逻辑代码，主要是通过给页面中每一项加一个序号，然后与ID相关联组成数组POST给listorder函数，在按照序号从大到小排序更新到数据库。

//此处为电影排序，其他模块类似，不赘述

public function listorder() {

$listorder = $\_POST['listorder'];//获取排序数组

$jumpUrl = $\_SERVER['HTTP\_REFERER'];//获取前一页面的url

$errors = array();

try {

if ($listorder) {

foreach ($listorder as $MovieId => $v) {

// 执行更新

$res = D("Movie")->updateMvListorderById($MovieId, $v);

if ($res === false) {

$errors[] = $MovieId;//更新失败的id存入数组

}

}

if ($errors) {

return show(0, '排序失败-' . implode(',', $errors), array('jump\_url' => $jumpUrl));

}

return show(1, '排序成功', array('jump\_url' => $jumpUrl));

}

}catch (Exception $e) {

return show(0, $e->getMessage());

}

return show(0,'获取排序数据失败',array('jump\_url' => $jumpUrl));

}

6.电影推送逻辑代码，主要是通过勾选所有需要推送的电影进行推送，通过勾选的ID获取电影信息在存入到电影推荐表中。

// 电影推送

public function push() {

$jumpUrl = $\_SERVER['HTTP\_REFERER'];

$movieId = $\_POST;

if(!$movieId || !is\_array($movieId)) {

return show(0, '请选择推荐电影ID进行推荐');

}

try {

//获取推送电影的ID数组

$Movie = D("Movie")->getMovieByMovieIdIn($movieId);

if (!$Movie) {

return show(0, '没有相关内容');

}

//遍历推送电影的ID数组，获取相关信息，并插入到数据库

foreach ($Movie as $new) {

$data = array(

'movie\_name' => $new['movie\_name'],

'pic' => $new['pic'],

'movie\_id' => $new['movie\_id'],

'status' => 1,

'grade' => $new['grade'],

'listorder' => $new['listorder'],

'push\_time' =>Date('Y-m-d H:i:s'),

);

$ret = D('MovieDetail')->find($data['movie\_id']);

$data['rank']=$ret['rank'];

//插入到数据库

$rank\_movie = D("RankMovie")->insert($data);

}

}catch(Exception $e) {

return show(0, $e->getMessage());

}

return show(1, '推荐成功',array('jump\_url'=>$jumpUrl));

}

7.批量删除逻辑代码，主要是通过勾选所有需要删除的电影进行删除，通过更改所有勾选的ID列表项的状态status为-1，表示该项已被删除，将不再前后台页面显示。

//歌曲批量删除，其他模块类似，不赘述

public function batchDel() {

$jumpUrl = $\_SERVER['HTTP\_REFERER'];

$musicId = $\_POST;

if(!$musicId || !is\_array($musicId)) {

return show(0, '请选择推荐歌曲ID进行批量删除');

}

try {//获取所有删除项的ID数组

$Music = D("Music")->getMusicIn($musicId);

if (!$Music) {

return show(0, '没有相关内容');

}

//遍历所有ID

foreach ($Music as $new) {

$data = array(

'id' => $new['id'],

);

$id = $data['id'];

if (!$id) {

return show(0, 'ID不存在');

}

//更新ID的状态为-1

$res = D("Music")->updateStatusById($id, -1);

}

}catch(Exception $e) {

return show(0, $e->getMessage());

}

return show(1, '删除成功',array('jump\_url'=>$jumpUrl));

}

1. 图片上传逻辑代码：分为无插件上传和通过插件kindupload实现图片上传

class ImageController extends CommonController {

private $\_uploadObj;

public function \_\_construct() {

}

//通过ajax上传图片

public function ajaxuploadimage() {

$upload = D("UploadImage"); //实例化图片上传函数

$res = $upload->imageUpload(); //调用图片上传函数实现上传

if($res===false) {

return show(0,'上传失败','');

}else{

return show(1,'上传成功',$res);

}

}

//通过kindupload插件上传图片

public function kindupload(){

$upload = D("UploadImage");

$res = $upload->upload();

if($res === false) {

return showKind(1,'上传失败');

}

return showKind(0,$res);

}

}

1. 数据库连接代码

<?php

return array(

'DB\_TYPE' => 'mysql', //数据库类型

'DB\_HOST' => 'localhost', //连接名称或ip

'DB\_USER' => 'root', // 用户

'DB\_PWD' => 'root', // 密码

'DB\_PORT' => 3306, // 端口号

'DB\_NAME' => 'movie', // 数据库名称

'DB\_CHARSET' => 'utf8', // 数据库编码格式

'DB\_PREFIX' =>'tb\_', // 数据库表前缀

);

1. thinkphp入口文件

// 应用入口文件

// 检测PHP环境

if(version\_compare(PHP\_VERSION,'5.3.0','<')) die('require PHP > 5.3.0 !')

// 开启调试模式 建议开发阶段开启 部署阶段注释或者设为false

define('APP\_DEBUG',True);

define('HTML\_PATH', './');

// 定义应用目录

define('APP\_PATH','./Application/');

define('DB\_FIELD\_CACHE',false);

define('HTML\_CACHE\_ON',false);

// 引入ThinkPHP入口文件

require './ThinkPHP/ThinkPHP.php';

第七章 结论

本系统综合运用了 sublime text3和 PHP WEB 网站建设技术制作本系统，关键技术是基于 PHP 开发电影推荐系统的设计与实现。本系统是应需求而开发的一个影视推荐系统，是一个能充分帮助大学生愉悦身心,丰富课外生活，增长阅历知识并提供一个新颖的信息交流平台。它主要由多个功能模块组成，其中前台页面包括：在线视听页面，上传页面，下载页面，影音信息查看页面，信息查询页面，会员操作页面，站长登录页面等；后台页面包括：上传日志管理页面，管理员对数据的管理页面，管理员对会员的管理页面以及目录管理页面。其中，前台主要实现在线视听、影视音乐上传、影视音乐下载、在线聊天和在线点歌等功能；后台主要用于管理员对影视音乐目录、数据信息和上传日志进行管理。本大学生在线影视系统采用关系数据库实现系统的功能，采用集中管理的方式来进行后台的数据管理，所有信息统一建设在一个数据库当中。普通用户及管理员的管理权限不同，普通用户只能进行浏览观看影音信息，点歌，管理个人信息等操作，但管理员可以对影音资源、影音目录、会员资料、最新上传等进行管理。该系统是基于 PHP 的相关技术来设计实现，还使用了一些html 脚本语言，与数据库进行连接采用 mysql方式，进行开发时也使网站界面尽量迎合大学生的审美标准，使网站具有界面友好，信息搜索方便快捷，系统安全稳定等特点。

参考文献

[1] 邓云，冯嘉礼.基于定性映射的影视推荐系统的应用与研究[J].现代计算机（普及版）上海海事大学信息工程学院,2013(1007-1423):3-7,11.

[2] 刘建国，周涛，汪秉宏.个性化推荐系统的研究进展[J].自然科学进展，2009,19(1):1-15.

[3] 门瑞.高质量个性化论文推荐系统研究[D].天津: 天津大学计算机科学与技术,2011:10-11.

[4] 赵丽嫚.一种新型的协同过滤推荐算法[D].南京：南京邮电大学计算机应用技术，2013:13-16.

[5] 黄倩.基于网页浏览行为的用户兴趣模型的研究[D].江西：东华大学信息与通信工程，2015:23-27.

[6] 孙海峰，甘明鑫，刘鑫，等.国外电影推荐系统网站研究与评述[J].计算机应用，2013,33（S2）:119-124.

[7] 陈天昊.互联网电影推荐方法的研究与实现[D].合肥:中国科技大学，2014:33-34.

[8] Gediminas Adomavicus,Alexander Tuzhilin. Toward the Next Generation of Rcommender Systems:A Survey of the State-of-the-Art and Possible Extensions [EI/OL].(2005-6)[2015-10].http://www.computer.oreg/csdl/trans/tk/2005/06/k0734 – abs.html.

[9] 李海龙. 大数据下电影推荐引擎的研究与实现[J].国家新闻出版广电总局电影数字节目管理中心,2016,7:2-3.

[10] 许海玲, 吴潇, 李晓东,等.互联网推荐系统比较研究[J].计算机应用研究, 2009,2:350-262.

[11] 吴颜，沈洁，顾天竺，等.协同过滤推荐系统中数据稀疏问题的解决[J].计算机应用研究, 2007,24(6):94-97.

[12] 王勇.大学生使用网络微视频的调查报告——以“长株潭”三地大学生为例[J].湖南工业大学学报:社会科学版,2013:15-17.

[13] 钱践.中国电影市场竞争性分析[J].新闻研究导刊,2012:20-22.

[14] 郭珈辰.电影推荐算法的研究与实现[D].长春：吉林大学，2015:55-58.

[15] 王守崑.个性化推荐算法的分类[J].程序员,2013:47-48.

[16] 邸亮.LDA模型在微博用户推荐中的应用[J].计算机工程,2016:28-31.

[17] 钱程.一种电影个性化推荐系统的研究与实现[J].计算机与数字工程,2015:40-41.  
[18] 江永胜.智能电视视频推荐系统的设计与实现[D].电子科技大学计算机软件与理论, 2015:23-24.

[19] 吴沧舟.基于MySQL数据库的优化[J].电子科技,2013:12-13.

[20] 聂芸.计算机软件测试方法概述[J].无线互联科技. 2015,9(1672-6944):59-60.  
[21] 张保军.软件产品测试流程管理分析[J].中国金融电脑. 2008,4(1001-0734):67-71.

成绩评定表

|  |  |  |
| --- | --- | --- |
| 学生小结 | 总之，这次毕业设计让我学习到很多。虽然结束了，但这只能是一个开始。今后今后作为程序员，要学习的规范，程序设计语言还有很多。怎样使自己从普通的PG升为SE，在计算机领域，要学的实在太多，仅大学生涯所学实在有限。我们只有对自己有了更高的要求，才能作为动力不断取得新的成绩! | |
| 教师评语 |  | |
| 成绩 | 平时成绩 | 综合成绩 |
|  |  |
| 指导教师（签字）： | |