Web 高级程序设计

姓　　名：陆宇韬 李彤兴　朱兆宁

指 导 教 师：　王 雷　　教授

专　　业：　　物联网

组员	分工
陆宇韬 201692458	后端
朱兆宁 201692473	前端
李彤兴 201692122	数据库设计

大连理工大学

Dalian University of Technology

目　　录

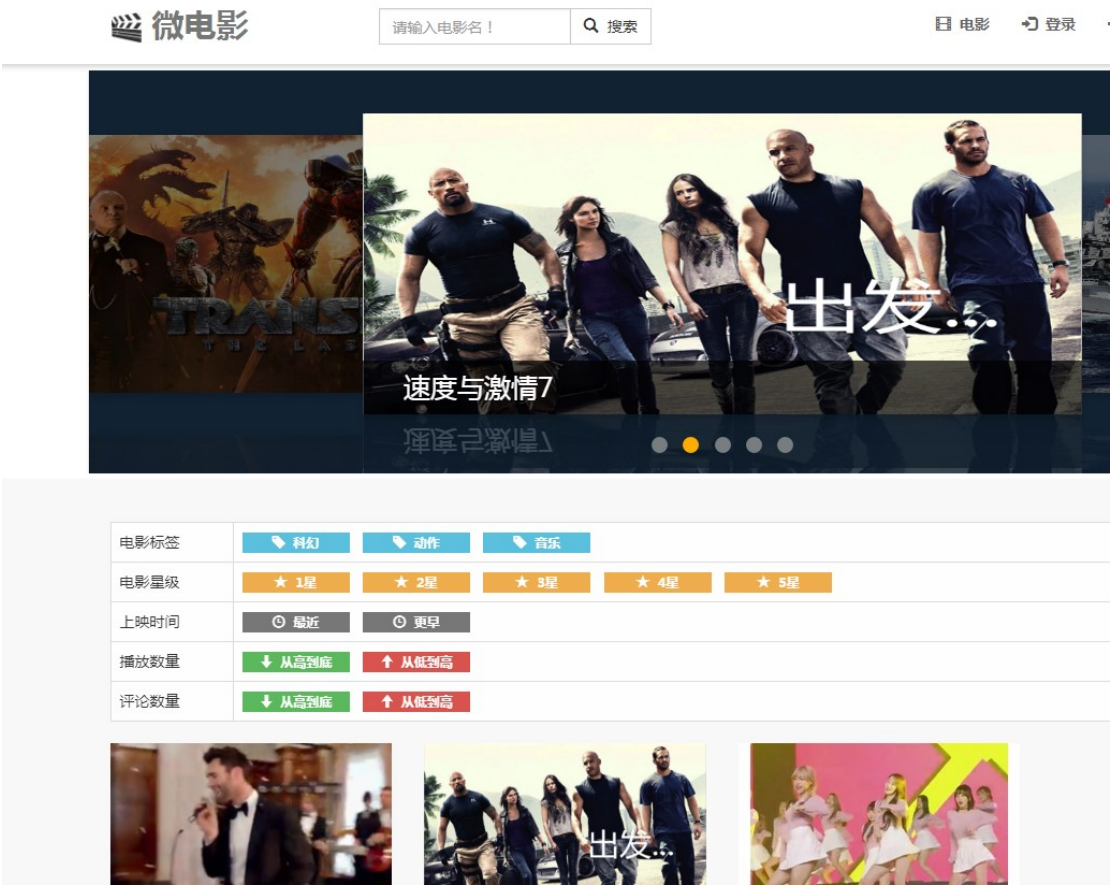
1 前端设计 3
1.1 Home 界面 3
1.2 登录界面 3
1.3 注册界面 4
1.4 搜索界面 5
1.5 电影界面 6
1.6 退出界面 8
1.7 会员界面 8
2 后台逻辑详解 10
2.1 后台整体设计 10
2.2 部分管理员主要功能代码介绍 11
2.3 部分会员功能介绍 13
3 数据库 14
3.1 前台 14
3.2 后台 19

1 前端设计

网站一共有 Home 界面、电影界面、登录界面、注册界面、搜索界面、退出界面、会员界面共七个前端界面。下面对每个界面逐一进行介绍。

1.1 Home 界面

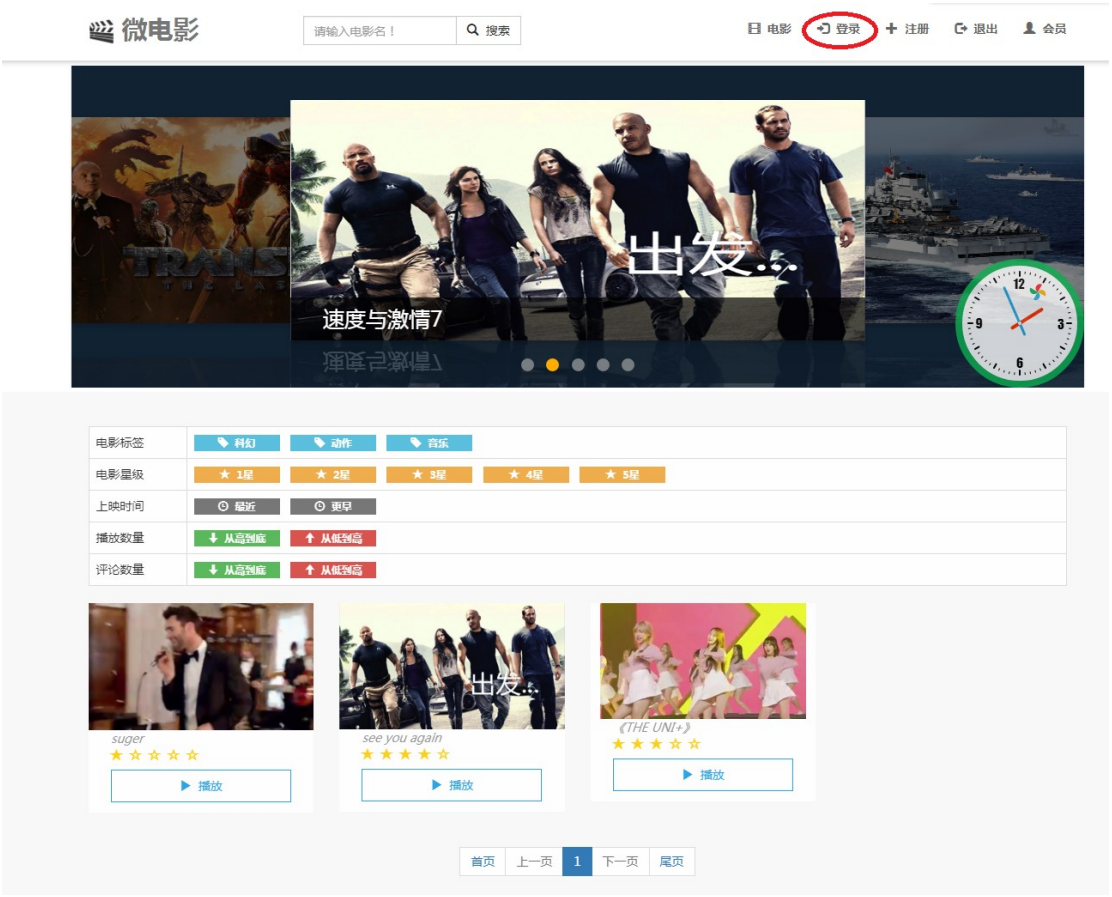
Home 界面为默认界面，打开网页 url 直接显示，由程序支持。界面中有六个跳转链接：搜索、电影、登录、注册、退出、会员。其中 Login 键跳往 Login 界面，进行已注册用户登录，Register 键跳往 Register 界面，进行新用户注册操作。



1.2 登录界面

如果访客为已注册用户，点击进入该页面输入账号和密码进行登录。界面由 3 个跳转链接：Home、登录、注册，两个输入框：用户名、密码，一个提交按钮：提交。用户输入账号密码后，将于数据库 users 表进行匹配，若匹配成功，

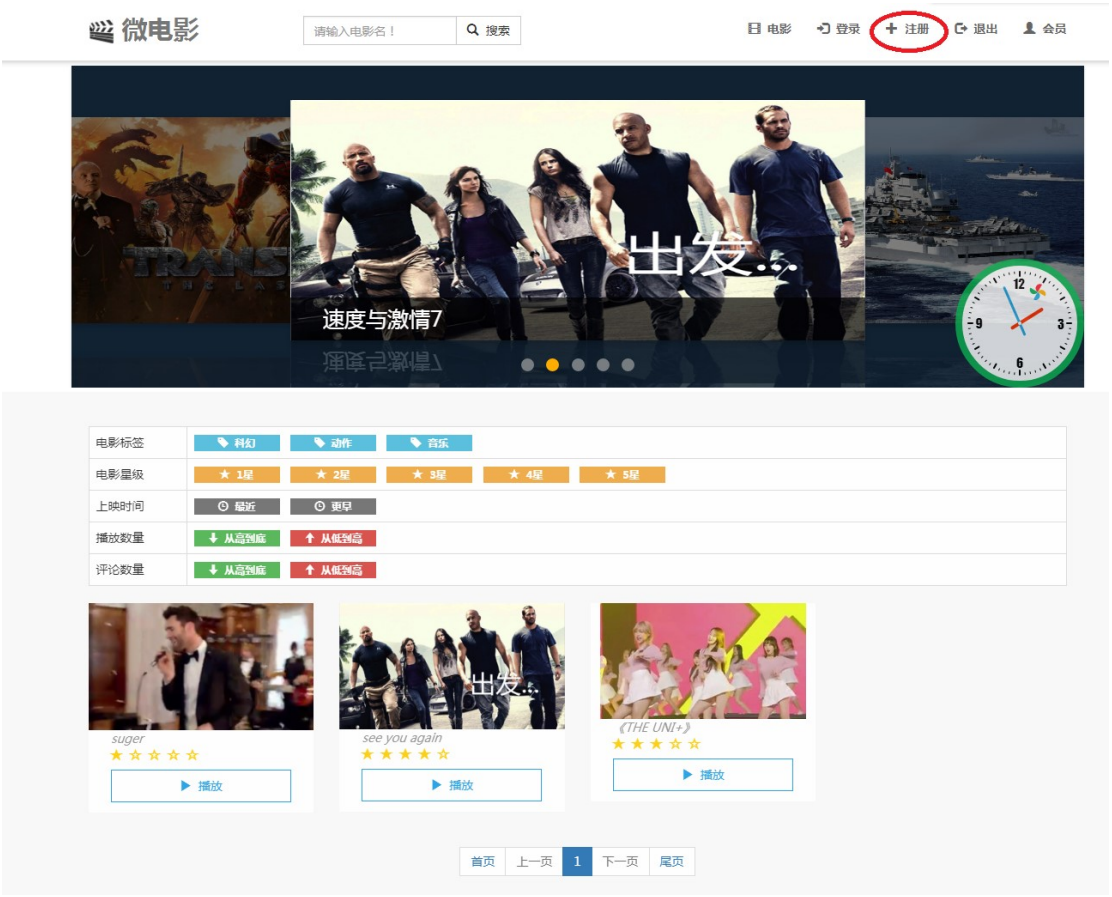
自动跳转到搜索界面，否则将返回错误。



1.3　注册界面

用于新用户的注册，新用户需要输入用户名和密码进行注册，界面有 3 个跳转链接：Home、登录、注册，两个输入框：用户名、密码、昵称、邮箱、手机号码、个性简介，一个按钮：提交。输入用户名和密码时，用户名不能与数据库

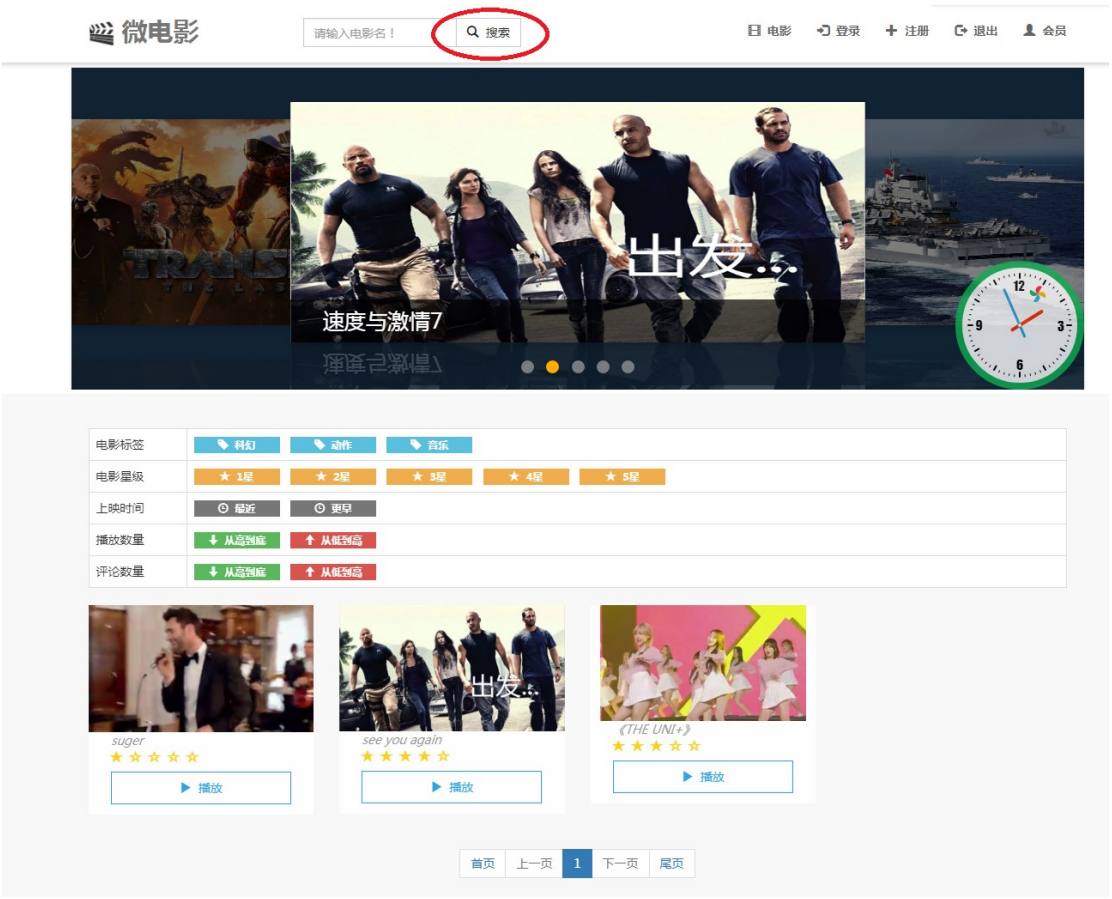
中已有的用户名重复，否则将会返回错误。注册成功后，将会跳转到搜索界面。



1.4　搜索界面

用于用户查询数据库中已有的视频和音频。由程序支持。界面由两部分组成：查询和导航。界面由 3 个跳转链接：Home、会员、退出，一个输入框：搜索，

一个提交按钮：提交。



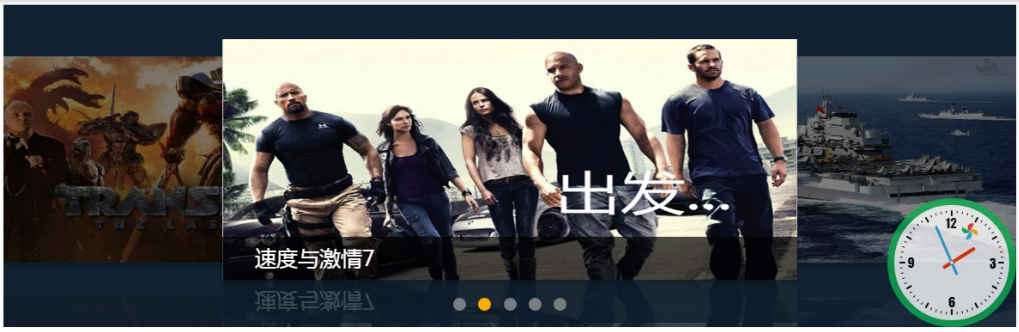
1.5　电影界面

用于显示按照排序的电影的编号、标题、封面、简介、星级、播放量、评论量、上映时间、上映地区、播放时间和影片时长，以及用于注册用户发表评论和添加收藏，界面由 3 个跳转链接：Home、会员、退出，一个输入框：影片评论，两个提交按钮：提交、收藏。

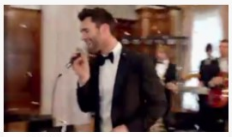
微电影

请输入电影名！ 搜索

电影 登录 注册 退出 会员



电影标签	科幻 动作 音乐
电影星级	★ 1星 ★ 2星 ★ 3星 ★ 4星 ★ 5星
上映时间	最近 更早
播放数量	从高到底 从低到高
评论数量	从高到底 从低到高



suger

★ ☆ ☆ ☆ ☆

播放

see you again

★ ★ ★ ★ ☆

播放

《THE UNI+》

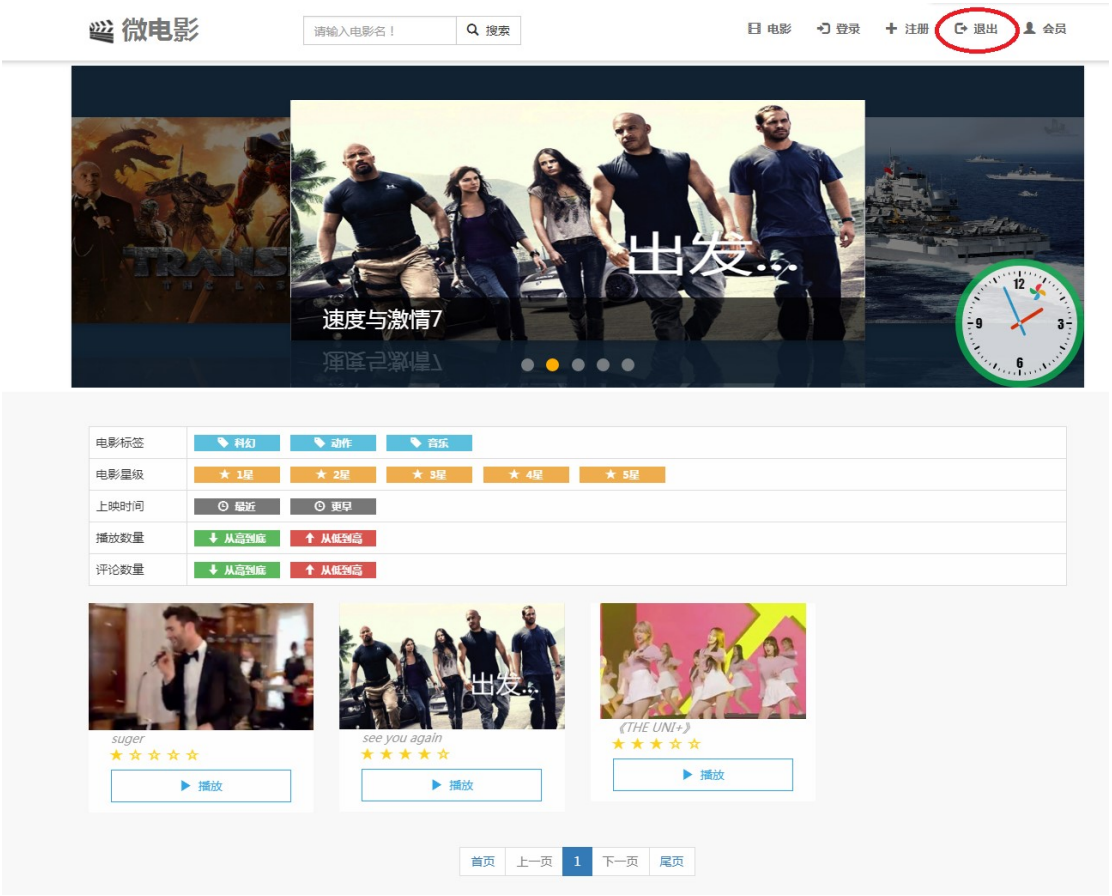
★ ★ ★ ☆ ☆

播放

首页 上一页 1 下一页 尾页

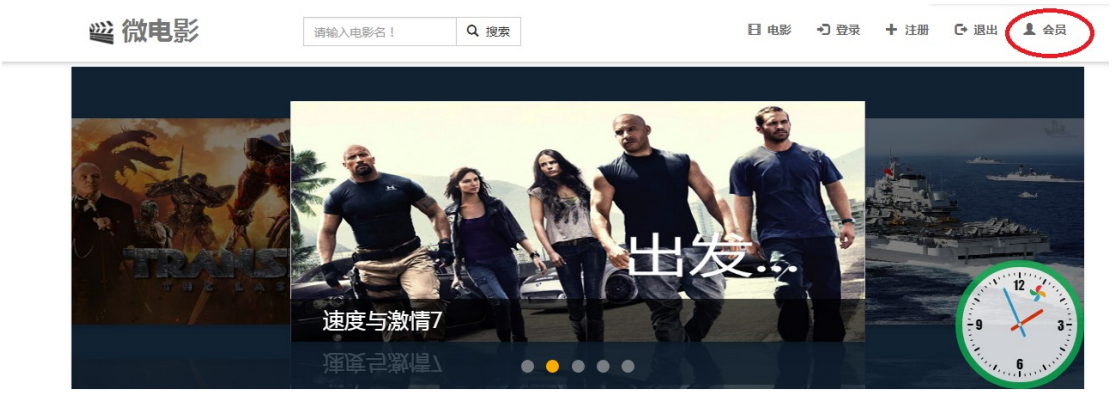
1.6　退出界面

用于用户的登出，用户点击进行登出。界面有 3 个跳转链接：Home、会员，1 个按钮：登出。

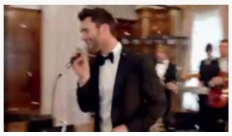


1.7　会员界面

用于管理员和注册会员的部分操作，管理员进入进行电影的增、删、改、查，以及其他管理操作。界面由两个跳转链接：Home、退出，十个提交按钮：标签管理、电源管理、预告管理、会员管理、评论管理、收藏管理、日志管理、权限管理、角色管理、管理员管理。



电影标签	科幻 动作 音乐
电影星级	★ 1星 ★ 2星 ★ 3星 ★ 4星 ★ 5星
上映时间	最近 更早
播放数量	从高到底 从低到高
评论数量	从高到底 从低到高



suger

播放

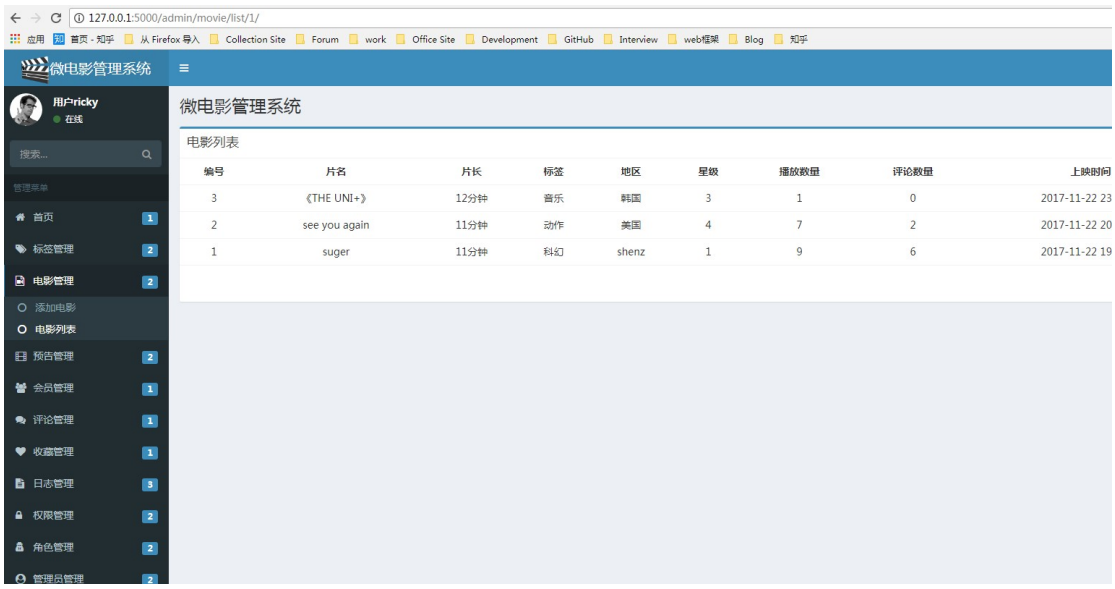
see you again

播放

《THE UNI+》

播放

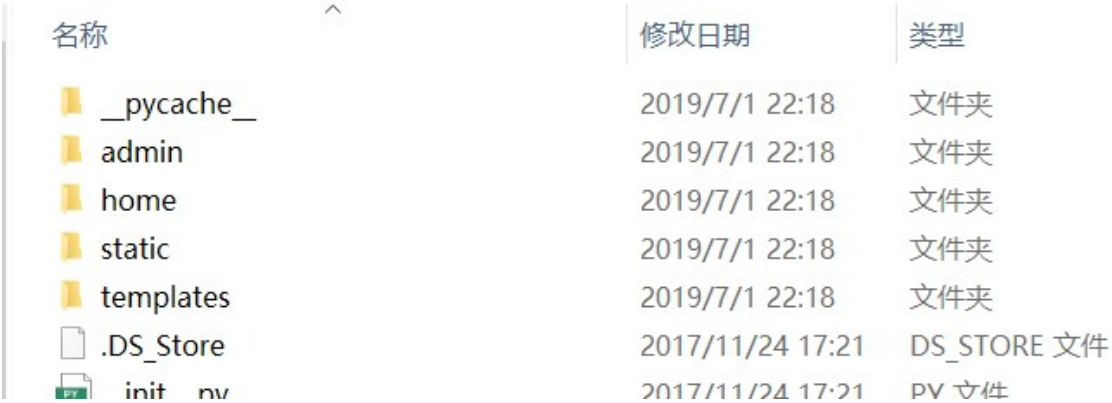
首页 上一页 1 下一页 尾页



2 后台逻辑详解

2.1 后台整体设计

系统使用了 flask 框架中的蓝图功能，将用户主要分为管理员和会员两类，分别对这两类人员的可以使用的功能分别设计。使用蓝图的好处是：
将不同的功能模块化；构建大型的应用；优化项目的结构；增强可读性，易于维护。
管理员的主要功能有：
管理员登录 / 修改密码 / 标签管理 / 电影管理 / 会员管理 / 评论管理 等
会员的主要功能有：
会员注册 / 修改密码 / 收藏影片 / 影片评论。
针对这两类角色的操作，分别在两个文件夹下创建了 admin 文件夹和 home 文件夹，两个文件下分别为各自角色的功能实现。



整体文件结构

目录下的 models.py 文件是记录着各个实体对象的属性，例如电影：

```python
# 电影
class Movie(db.Model):
    __tablename__ = "movie"
    id = db.Column(db.Integer, primary_key=True)  # 编号
    title = db.Column(db.String(255), unique=True)  # 标题
    url = db.Column(db.String(255), unique=True)  # 地址
    info = db.Column(db.Text)  # 简介
    logo = db.Column(db.String(255), unique=True)  # 封面
    star = db.Column(db.SmallInteger)  # 星级
    playnum = db.Column(db.BigInteger)  # 播放量
    commentnum = db.Column(db.BigInteger)  # 评论量
    tag_id = db.Column(db.Integer, db.ForeignKey('tag.id'))  # 所属标签
    area = db.Column(db.String(255))  # 上映地区
    release_time = db.Column(db.Date)  # 上映时间
    length = db.Column(db.String(100))  # 播放时间
    addtime = db.Column(db.DateTime, index=True, default=datetime.now)  # 添加时间
    comments = db.relationship("Comment", backref='movie')  # 评论外键关系关联
    moviecols = db.relationship("Moviecol", backref='movie')  # 收藏外键关系关联

    def __repr__(self):
        return "<Movie %r>" % self.title
```

目录下的_init_.py 文件为 flask 框架的一些配置：

```python
from flask_sqlalchemy import SQLAlchemy
from flask_redis import FlaskRedis
import pymysql
import os

app = Flask(__name__)
app.config["SQLALCHEMY_DATABASE_URI"] = "mysql+pymysql://root:root@localhost/movie"
app.config["SQLALCHEMY_TRACK_MODIFICATIONS"] = True
app.config["SECRET_KEY"] = 'af2fad8cfe1f4c5fac4aa5edf6fcc8f3'
app.config["REDIS_URL"] = "redis://192.168.4.1:6379/0"
app.config["UP_DIR"] = os.path.join(os.path.abspath(os.path.dirname(__file__)), "static/uploads/")  #
app.config["FC_DIR"] = os.path.join(os.path.abspath(os.path.dirname(__file__)), "static/uploads/users/"
app.debug = True
db = SQLAlchemy(app)
rd = FlaskRedis(app)

from app.home import home as home_blueprint
from app.admin import admin as admin_blueprint

app.register_blueprint(home_blueprint)
app.register_blueprint(admin_blueprint, url_prefix="/admin")


@app.errorhandler(404)
def page_not_found(error):
    return render_template("home/404.html"), 404

```

2.2 部分管理员主要功能代码介绍

管理员登陆：

```python
# 登录
@admin.route("/login/", methods=["GET", "POST"])
def login():
    form = LoginForm()
    if form.validate_on_submit():
        data = form.data
        admin = Admin.query.filter_by(name=data["account"]).firs
        if not admin.check_pwd(data["pwd"]):
            flash("密码错误！", 'err')
            return redirect(url_for("admin.login"))
        session["admin"] = data["account"]
        session["admin_id"] = admin.id
        # 管理员操作日志
        adminlog = Adminlog(
            admin_id=admin.id,
            ip=request.remote_addr,
        )
```

管理员注册之后可以登陆，服务器端使用 session 记录管理员登陆信息。

添加电影：

```python
@admin.route("/movie/add/", methods=["GET", "POST"])
@admin_login_req
# @admin_auth
def movie_add():
    form = MovieForm()
    if form.validate_on_submit():
        data = form.data
        file_url = secure_filename(form.url.data.filename)
        file_logo = secure_filename(form.logo.data.filename)
        if not os.path.exists(app.config["UP_DIR"]):
            os.makedirs(app.config["UP_DIR"])
            os.chmod(app.config["UP_DIR"], "rw")
        url = change_filename(file_url)
        logo = change_filename(file_logo)
        form.url.data.save(app.config["UP_DIR"] + url)
        form.logo.data.save(app.config["UP_DIR"] + logo)
        movie = Movie(
            title=data["title"],
            url=url,
            info=data["info"],
            logo=logo,
            star=int(data["star"]),
            playnum=0,
            commentnum=0,
            tag_id=int(data["tag_id"]),
            area=data["area"],
            release_time=data["release_time"],
            length=data["length"]
        )
        db.session.add(movie)
        db.session.commit()
        flash("添加电影成功! ", "ok")
        return redirect(url_for('admin.movie_add'))
    return render_template("admin/movie_add.html", form=form)
```

在数据库中添加电影信息。

2.3 部分会员功能介绍

会员注册：

会员第一次使用此系统需要进行注册，在注册时指定用户名，密码，手机号码等信息，其中，用户名和手机号码是唯一的注册后向数据库中写入此用户信息，用户使用用户名和密码登陆，使用手机号码带找回密码。

```python
# 会员注册
@home.route("/regist/", methods=["GET", "POST"])
def regist():
    form = RegistForm()
    if form.validate_on_submit():
        data = form.data
        user = User(
            name=data["name"],
            email=data["email"],
            phone=data["phone"],
            pwd=generate_password_hash(data["pwd"]),
            uuid=uuid.uuid4().hex
        )
        db.session.add(user)
        db.session.commit()
        flash("注册成功！", "ok")
    return render_template("home/regist.html", form=form)
```

电影搜索：

用户通过电影标题来搜索电影

```python
# 搜索
@home.route("/search/<int:page>/")
def search(page=None):
    if page is None:
        page = 1
    key = request.args.get("key", "")
    movie_count = Movie.query.filter(
        Movie.title.ilike('%' + key + '%')
    ).count()
    page_data = Movie.query.filter(
        Movie.title.ilike('%' + key + '%')
    ).order_by(
        Movie.addtime.desc()
    ).paginate(page=page, per_page=10)
    page_data.key = key
    return render_template("home/search.html", movie_count=movie_count, key=key, page_data=page_data)
```

3 数据库

数据库作为开发中的重点，也付出了很长时间去设计数据库。设计好了数据库，那么我们开发就成功了一半，可以看出数据库有多重要了。

3.1 前台

首先，对于前台的模型，设计了以下几个表：

- 会员表（user）
- 会员登录日志（userlog）
- 标签表（tag）
- 电影表（movie）
- 上映预告（preview）
- 评论（comment）

- 电影收藏（moviecol）

其中，会员表包含的元素及关系为

会员表（user）	
id	编号，整型，主键，自动递增
name	昵称，字符串型，唯一
pwd	密码，字符串型
email	邮箱，字符串型，唯一
phone	手机号码，字符串型，唯一
info	个性简介，文本型
face	头像，字符串型，唯一
addtime	添加时间，日期时间类型，默认当前十年
uuid	唯一标识符，字符串型，唯一
uerlog	关联会员登录日志模型
comments	关联评论模型

会员表（user）	
moviecols	关联收藏模型

会员登录日志表（userlog）中的元素及关系为

会员登录日志表（userlog）	
id	编号，整型，主键，自动递增
user_id	所属会员 ID，整型，关联 user 表的 id 字段
ip	IP 地址，字符串型
addtime	添加时间，日期时间类型，默认为当前的时间

标签表（tag）包含的元素及关系为

标签表（tag）	
id	编号，整型，主键，自动递增
name	标题，字符串型，唯一
addtime	添加时间，日期时间类型，默认为当前的时间

movies	关联电影模型

电影表（movie）包含的元素及关系为

电影表(movie)	
id	编号，整型，主键，自动递增
title	标题，字符串型，唯一
url	地址，字符串型，唯一
info	简介，文本型
logo	封面，字符串型，唯一
star	星型，小整型
playnum	播放量，大整型
commentnum	评论量，大整型
tag_id	所属标签 ID,整型，关联 tag 表的 id 字段
area	上映地区，字符串类型

release_time	上映时间，日期类型
length	播放时间，字符串整型
addtime	添加时间，日期时间型，默认单前时间
comments	关联评论模型
moviecols	关联电影收藏模型

上映预告（preview）表中的元素及关系为

id	编号，整型，主键，自动递增
title	标题，字符串型，唯一
logo	封面，字符串型，唯一
addtime	添加时间，日期时间型，默认为当前时间

评论（comment）中元素及关系为

评论（comment）	
id	编号，整型，主键，自动递增

content	内容，文本型
movie_id	所属电影 ID，整型，关联 movie 表的 id 地段
user_id	所属用户 ID，整型，关联 user 表的 id 字段
addtime	添加时间，日期时间型，默认为当前时间

电影收藏（moviecol）中元素及关系为

电影收藏（moviecol）	
id	编号，整型，主键，自动递增
movie_id	所属电影 ID，整型，关联 movie 表的 id 地段
user_id	所属用户 ID，整型，关联 user 表的 id 字段
addtime	添加时间，日期时间型，默认为当前时间

3.2 后台

对于后台的模型，设计了以下几个表

- 权限表（auth）
- 角色表（role）
- 管理员表（admin）
- 管理员登录日志（adminlog）

- 操作日志（oplog）

权限（auth）表中元素及关系为

权限表（auth）	
id	编号，整型，主键，自动递增
name	名称，字符串型，唯一
url	地址，字符串型，唯一
addtime	添加时间，日期时间类型

角色表（role）中元素及关系为

角色表（role）	
id	编号，整型，主键，自动递增
name	名称，字符串型，唯一
auths	角色权限列表，字符串型
addtime	添加时间，日期时间类型，默认为当前时间

admin	关联 admin 模型

管理员表（admin）中元素及关系为

管理员表（admin）	
id	编号，整型，主键，自动递增
name	管理员账号，字符串型，唯一
pwd	管理员密码，字符串型
is_super	是否为超级管理员，小整型
role_id	所属角色 ID，模型，关联 role 表的 id 字段
addtime	添加时间，日期时间类型，默认为当前时间
adminlogs	关联 adminlog 模型
oplogs	关联 oplog 模型

管理员登录日志（adminlog）中元素及关系为

管理员登录日志（adminlog）	

id	编号，整型，主键，自动递增
adminlog	所属管理员 id,整型，关联 admin 的 id 字段
ip	登录 IP,字符串型
addtime	添加时间，日期时间类型，默认为当前时间

操作日志（oplog）表中元素及关系为

操作日志（oplog）	
id	编号，整型，主键，自动递增
adminlog	所属管理员 ID，整型，管理 admin 的 id 字段
ip	操作 IP,字符串型
reason	操作原因，字符串型
addtime	添加时间，日期时间类型，默认为当前时间

设计好数据库，项目已经完成大半。