数据库五张表，电影信息表，电影主题表，历史记录表，用户信息表，主题表

电影主题表参考主题表和电影信息表 多对多关联，电影主题表为中间表，用于Mybatis的多对多关联

历史记录表参考用户信息表和电影信息表，并且添加自己的浏览时间，历史记录表用于保存用户的浏览记录，历史记录表作为中间表用来进行mybatis多对多关联

持久层 edu.lnut.po包含六个POJO对象，

1、有电影信息表FilmInfo用作查找电影的信息，其中包含电影主题的POJO对象，用于存放电影的类型信息，不用时值为空

2、电影主题表FilmTheme用作查找这个电影的所有主题（电影主题可能有多个），

3、历史记录表对象用于插入用户观影的历史记录，

4、用户浏览记录对象（Records）用于查询当前用户所观影的全部记录（一个用户可能观看多部电影，一对多关联映射），

5、主题对象（在初始化视图时进行主题读取，包含一个FilmInfo对象的List列表，用于进行一对多关联，查找某一主题的所有对象）和用户对象对用户表进行维护，包括插入和查找。

Dao层包括五个Dao对象；本层的开发使用基于MapperScannerConfigurer的整合 只需在edu.lnut.dao编写与Dao相同名称的xml映射文件，系统会自动生成映射器。

1、ThemeDao对象有两个方法，selectThemeList()方法用于查找所有主题并返回一个包含Theme对象的List，使用了动态Sql的<where>和<if>元素，当需要查找某个类型时，按照类型名查找，当没有给定要查找的类型时查找全部类型的电影。

。selectFilmInfoList()方法用于根据给定的主题名称查找当前主题的所有电影，

2、FilmInfoDao对象用于查找电影信息表，返回一个包含电影信息POJO对象的List用来根据电影编号进行查找，使用了多表连接，查找当前电影编号的所有的主题名等信息将主题信息封装在List中

3、UserRegisterDao对象用于进行用户注册信息的维护，接收一个User对象，按照User的各个属性进行insert语句的编写。

4、UserDao对象维护User表的内容，findUserFilm()方法根据用户名查询返回User对象，findUserEmail()方法查询用户注册的邮箱是否重复。

5、RecordsDao对象用于对用户的历史记录进行操作

findRecords()方法用于根据用户的ID查询用户的浏览记录，使用多表查询，返回一个Records对象，包含当前用户的浏览记录，其中包含一个List < FilmInfo >对象用来将用户观看的所有电影显示。

insertRecords()方法用于插入用户浏览的记录，通过HistoryRecords对象的各个参数进行insert插入。

服务层service开发；本层所有的UserDao对象创建使用基于annotation的装配方式;使用包路径扫描的方式扫描指定包下的所有Bean

包含十个类，分别为5个接口和5个接口对应的实现类。

分别为：

1、FilmInfoService接口和本接口的实现类FileInfoServiceImpl，实现类中包含一个FilmInfoDao对象，selectFilmInfoList()方法调用FilmInfoDao的selectFilmInfoList()方法返回这部电影的类型，返回值为List<FilmInfo>其中FilmInfo中包含Theme对象列表，为当前对象的所有主题。

2、ThemeService接口和ThemeServiceImpl实现类，本类包含一个ThemeDao对象，接口实现类的findThemeList()方法调用ThemeDao对象的selectThemeList()方法返回所有的主题,用于初始化视图时从数据库中查找所有的电影类型. findFilmThemeList()方法调用ThemeDao类的selectFilmInfoList()方法根据动态Sql判断查询全部还是按照类型查询

3、UserRecordService接口和l接口的实现类UserRecordServiceImp;包含一个RecordsDao对象,本实现类的findUserRecord()方法调用

RecordsDao的findRecords()方法,按照用户Id查找当前用户的用户历史记录接受一个Records列表对象.

本类insertUserRecord()方法调用RecordsDao的insertRecords()方法,通过HistoryRecords的POJO插入用户的浏览记录.

4、InsertUserService接口和实现类InsertUserServiceImpl,包含一个UserRegisterDao对象,本类的insertUser方法调用UserRegisterDao的insertUser(user);方法,接受一个User的持久层对象,按照对象的参数添加用户.

UserService接口和实现类UserServiceImpl,包含一个UserDao对象,本类的findUserInfo方法调用UserDao.findUserFilm(userName);方法用于根据用户名查找用户User持久层对象.

findUserEmail方法通过调用userDao.findUserEmail(userEmail);方法根据用户邮箱查找用户User持久层对象.

控制层@Controller；spring mvc配置

控制层的实现类FilmController包含的属性：

ThemeService themeService;

FilmInfoService filmInfoService;

UserService userService;// 查找用户信息

UserRecordService userRecordService;

InsertUserService insertUserService;

为服务层的各种类，采用基于annotation的装配方式自动装配。配置前端核心控制器，将所有请求拦截到本类中。本类中方法见代码。

Spring mvc配置：使用了基于注解的控制器实现方案，没有使用继承接口的控制器方案，使用基于注解的控制器可以同时处理多个请求动作，更加简单，只需要在spring-mvc.xml中配置扫描器。

在spring4.0以后无需配置处理器映射器，处理器适配器和视图解析器

<!-- 配置扫描器 -->

<context:component-scan base-package="edu.lnut.controller" />

1、配置前端核心控制器 web.xml中

2、无需显式配置处理器映射器，处理器适配器和视图解析器

3、配置视图解析器解析视图，实现数据和视图的完全解耦

AOP编程 使用了基于注解的声明式AspectJ实现AOP编程

<aop:aspectj-autoproxy />

1、HistoryAspect切面。配置切入点"execution(\* edu.lnut.service.impl.FileInfoServiceImpl.selectFilmInfoList(..))"

//拦截所有返回值类型，[edu.lnut.service.impl.FileInfoServiceImpl下的selectFilmInfoList方法](mailto:edu.lnut.service.impl.FileInfoServiceImpl下的selectFilmInfoList方法使用一个后置通知@AfterReturning)包含一个UserRecordService服务层对象，使用注解自动装配

使用一个后置通知@AfterReturning在用户登录后且单击电影时代表用户进行了观影操作；最后调用Service层的插入数据方法,在打开播放后进行插入数据将电影信息，用户信息，当前时间写入浏览记录表。

2、InsertAspect切面

拦截所有返回值类型，edu.lnut.service.impl.InsertUserServiceImpl下的insertUser方法

定义一个环绕通知 @Around在用户提交注册时验证数据库中是否包含该用户的用户名，如果包含则不执行注册操作，并且发送错误响应到视图，提示用户当前用户已被注册。

拦截器

LoginInterceptor登录拦截器类：要想访问电影列表，用户必须登录

HistoryInterceptor历史记录拦截器类：要想访问当前用户的历史记录，用户必须登录

在spring-mvc中配置拦截器

<!-- 配置拦截器 -->

<mvc:interceptors>

<mvc:interceptor>

<mvc:mapping path="/context/\*\*" />

<bean class="edu.lnut.interceptor.LoginInterceptor" />

</mvc:interceptor>

<mvc:interceptor>

<mvc:mapping path="/history/\*\*" />

<bean class="edu.lnut.interceptor.HistoryInterceptor" />

</mvc:interceptor>

</mvc:interceptors>

视图（jsp）文件

Context.jsp 电影播放器视图, 拦截器,登陆后可访问

History.jsp 用户历史记录视图,拦截器,登陆后可访问

Index.jsp 首页视图

Login.jsp 用户登录视图

Register.jsp 注册视图

Json数据交互

配置注解驱动

<mvc:annotation-driven />

提供给控制器对Json功能的支持

ajaxUser方法

@ResponseBody//这个注解通常使用在控制层(controller)的方法上，其作用是将方法的返回值以特定的格式写入到response的body区域，进而将数据返回给客户端。

前端使用$.ajax()获取Json数据