互联网电影购票系统

软件设计文档

目录

一．技术选型……3

1.1　技术选型表……3

1.2　技术选型理由……4

二．架构设计……6

2.1　用例图……6

2.2　领域模型……7

2.3　数据库模型……7

2.4　功能建模……8

三．模块划分……10

四．使用的软件设计技术

一． 技术选型

1.1 该项目的技术选型表如下：

项目	web app
1 终端支持	□ PC □ Pad □ Phone
1.1 开发语言框架	□ HTML 5 □ CSS 3 □ JavaScript
1.2 响应式布局框架	□ BootStrip
1.3 传感器	□ GPS
2 服务器端支持	
2.1 语言	□ Java □ JavaScript
2.2 web 框架	□ Struts + Spring □ Node.js □ Django
2.3 ORM 框架	□ Hibernate
2.4 关系数据库	□ MySQL
2.5 数据缓存（非关系）	□ NoSQL □ Redis □ MangoDB
2.7 负载均衡机制	□ Nginx
2.8 消息中间件	□ ZeroMQ □ RabbitMQ □ ActiveMQ
2.9 其他第三方组件	□ XX 地图 API
3 开发平台与工具	

3.1 IDE	□
3.2 集成与测试	□
3.3 源代码管理	□ Github

1.2 技术选型理由：

a) 技术语言：

i. HTML5

- 语言特性：HTML 5 赋予了网页更好的意义和结构，也有这更加丰富的标签，随着对 RDFa，微数据和微格式等方面的支持，构建了对程序、对用户都更有价值的数据驱动的 web。
- 本地存储：基于 HTML 5 开发的网页 APP 拥有更短的启动时间，更快的联网速度，
- 设备兼容：HTML 5 为网页应用开发者们提供了更多功能上的优化选择，带来了更多体验功能的优势。HTML 5 提供了前所未有的数据与应用接入开发，使得外部应用可以直接与浏览器内部的数据直接相连。
- 连接特性：HTML 5 具有更有效的连接工作的效率，使得能实现基于页面的实时聊天，更快速的网页游戏体验，更优化的在线聊天。另外，HTML 5 拥有更有效的服务器推送技术，如 Server-Sent Event 和 WebSockets 就是其中的两个特性，这两个特性能够帮助我们实现服务器将数据推送到客户端。
- 网页多媒体性 :支持网页端的 Audio、Video 等多媒体功能 ,与网站自带 APPS, 摄像头，影音功能相得益彰。

ii. CSS3

- CSS 是指层叠样式表，在网页制作时采用 css 技术，可以有效地对网页的布局、字体、颜色、背景和其他效果实现更加精确的控制。
- 它定义了如何显示 HTML 元素，通常储存在样式表中
- 外部样式表极大地提高了工作效率
- 多个样式表定义可层叠为一

iii. JavaScript

- JavaScript 是属于网络的脚本语言，被数以万计的网页用来改进设计、验证表单、检测浏览器、创建 cookie，以及其他更多的应用。
- JavaScript 是一种动态类型、弱类型、基于原型的语言，内置支持类型，它的解释器被称为 JavaScript 引擎，为浏览器的一部分，广泛用于客户端的脚本语言。
- JavaScript 是因特网上最流行的脚本语言，也很容易使用
- 可用于 HTML 和 web ,更可广泛用于服务器 ,PC ,平板或者智能手机等设备。
- JavaScript 是一种轻量级的编程语言，插入 HTML 页面后，可由所有的现代浏览器执行

b) 响应式布局框架：

i. 响应式布局可以为不同终端的用户提供更加舒适的界面和更好的用户体验，而且随着大屏幕移动设备的普及，趋势都是要用响应式布局。

ii. 布局网站可以很好地解决不同设备的屏幕分辨率 ,那么在网站设计中就不能使用绝对定位，在设计的时候就要考虑可读性，区域面积以及再不同设备下的行

为。

iii. 另外，响应式的基础是 HTML 5 与 CSS 3 的搭配，需要熟练掌握 HTML 5 以及 CSS 3.

c) 关系数据库

i. 关系数据库，是创建在模型基础上的数据库，运用 mysql 来实现

ii. Mysql 支持多线程，能充分利用 CPU 资源，是优化的 SQL 查询算法，能够有效地提高查询速度。

d) 负载均衡机制：Nginx

i. 是一个高性能的 HTTP 和反向代理的服务器及电子邮件代理服务器。

ii. 相对于 Apache、lighttpd 具有占有内存少、稳定性高等优势。

iii. 具有无缓存的反向代理加速、简单的负载均衡和容错的优点。

iv. 另外 Nginx 中，整体采用模块化是一个重大特点，甚至 http 服务器核心功能也是一个模块。

e) Spring MVC

i. 方便解耦，简化开发。通过 Spring 提供的 IoC 容器，我们可以将对象之间的依赖关系交由 Spring 进行控制，避免硬编码所造成的过度程序耦合。有了 Spring，用户不必再为单实例模式类、属性文件解析等这些很底层的需求编写代码，可以更专注于上层的应用。

ii. AOP 编程的支持。通过 Spring 提供的 AOP 功能，方便进行面向切面的编程，许多不容易用传统 OOP 实现的功能可以通过 AOP 轻松应付。

iii. 声明事物的支持。在 Spring 中，我们可以从单调烦闷的事务管理代码中解脱出来，通过声明式方式灵活地进行事务的管理，提高开发效率和质量。

iv. 方便程序的测试。可以用非容器依赖的编程方式进行几乎所有的测试工作，在 Spring 里，测试不再是昂贵的操作，而是随手可做的事情。例如：Spring 对 Junit4 支持，可以通过注解方便的测试 Spring 程序。

v. 方便集成各种优秀框架。Spring 不排斥各种优秀的开源框架，相反，Spring 可以降低各种框架的使用难度，Spring 提供了对各种优秀框架（如 Struts,Hibernate、Hessian、Quartz）等的直接支持。

vi. 降低 Java EE API 的使用难度。Spring 对很多难用的 Java EE API（如 JDBC，JavaMail，远程调用等）提供了一个薄薄的封装层，通过 Spring 的简易封装，这些 Java EE API 的使用难度大为降低。

f) Hibernate

i.

ii. 对象化。hibernate 可以让开发人员以面相对象的思想来操作数据库。jdbc 只能通过 SQL 语句将元数据传送给数据库，进行数据操作。而 hibernate 可以在底层对元数据和对象进行转化，使得开发者只用面向对象的方式来存取数据即可。

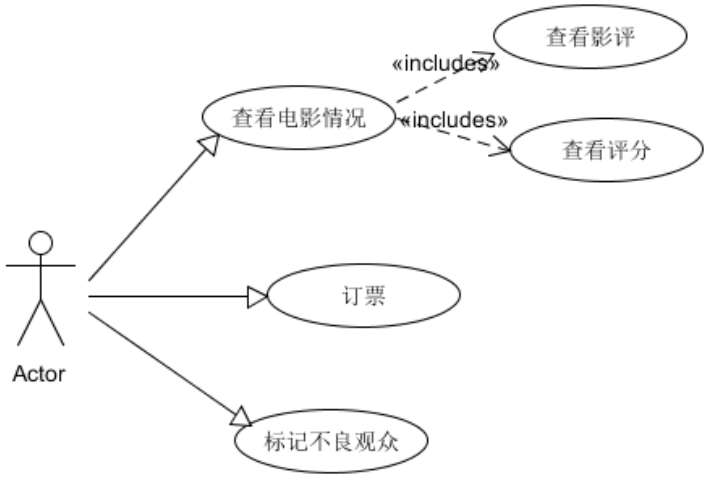
iii. 更好的移植性。hibernate 使用 xml 或 JPA 的配置以及数据库方言等等的机制，使得 hibernate 具有更好的移植性，对于不同的数据库，开发者只需要使用相同的数据操作即可，无需关心数据库之间的差异。而直接使用 JDBC 就不得不考虑数据库差异的问题。

iv. 开发效率高。hibernate 提供了大量的封装（这也是它最大的缺点），很多数据操作以及关联关系等都被封装的很好，开发者不需写大量的 sql 语句，这就极大的提高了开发者的开发效率。

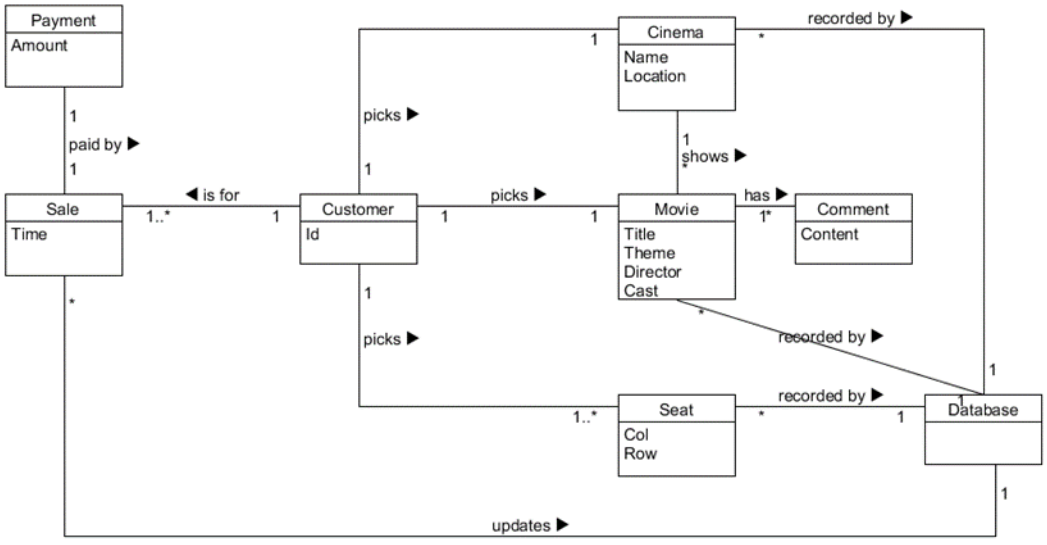
v. 缓存机制的使用。hibernate 提供了缓存机制（session 缓存，二级缓存，查询缓存），对于那些改动不大且经常使用的数据，可以将它们放到缓存中，不必在每次使用时都去查询数据库，缓存机制对提升性能大有裨益。

二． 架构设计

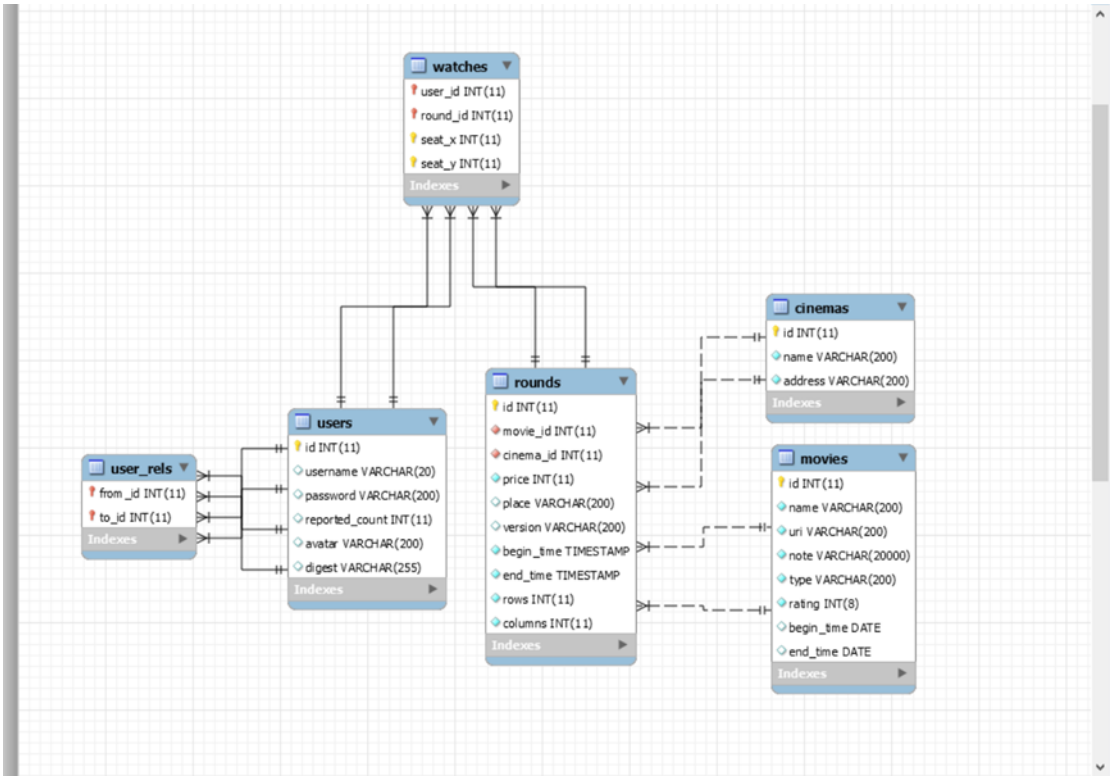
2.1 用例图



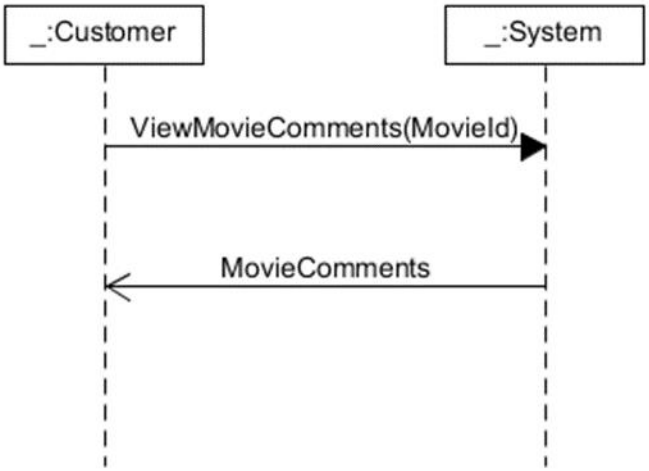
2.2 领域模型

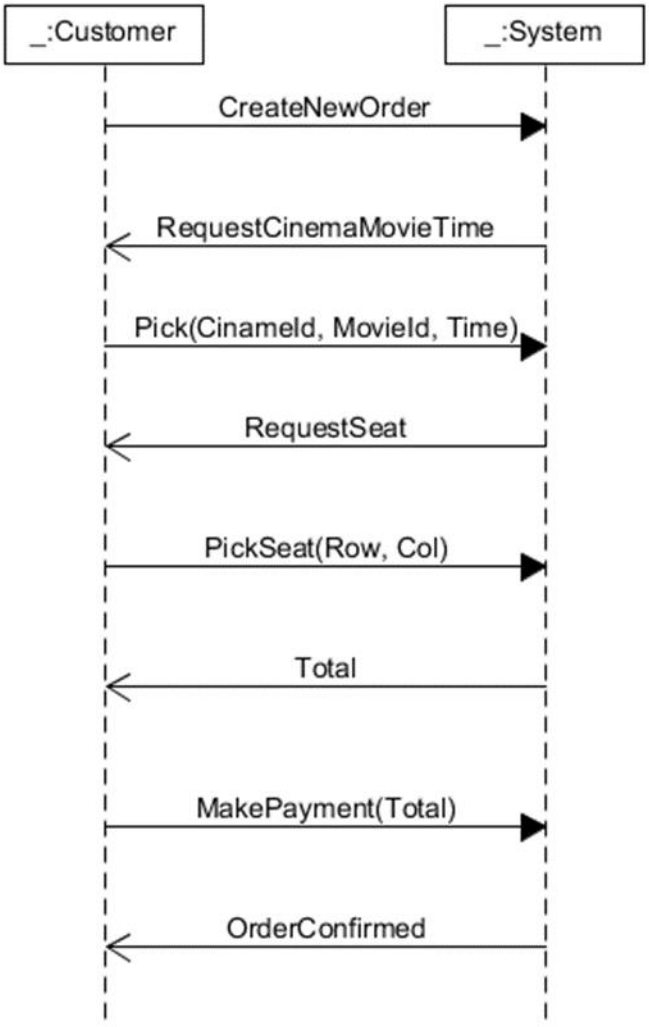


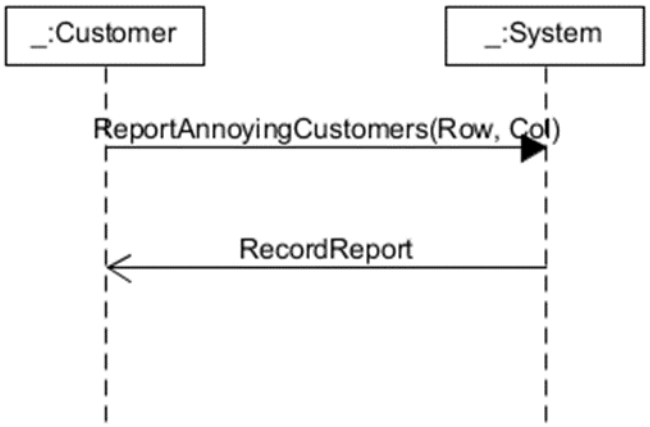
2.3 数据库模型


watches
user_id INT(11)
round_id INT(11)
seat_x INT(11)
seat_y INT(11)
Indexes
cinemas
id INT(11)
name VARCHAR(200)
address VARCHAR(200)
Indexes
rounds
id INT(11)
movie_id INT(11)
cinema_id INT(11)
price INT(11)
place VARCHAR(200)
version VARCHAR(200)
begin_time TIMESTAMP
end_time TIMESTAMP
rows INT(11)
columns INT(11)
Indexes
users
id INT(11)
username VARCHAR(20)
password VARCHAR(200)
reported_count INT(11)
avatar VARCHAR(200)
digest VARCHAR(255)
Indexes
user_rels
from_id INT(11)
to_id INT(11)
Indexes
movies
id INT(11)
name VARCHAR(200)
uri VARCHAR(200)
note VARCHAR(20000)
type VARCHAR(200)
rating INT(8)
begin_time DATE
end_time DATE
Indexes

2.4 功能建模


_:Customer
_:System
ViewMovieComments(MovieId)
MovieComments


_:Customer
_:System
CreateNewOrder
RequestCinemaMovieTime
Pick(CinameId, MovieId, Time)
RequestSeat
PickSeat(Row, Col)
Total
MakePayment(Total)
OrderConfirmed



①操作：CreatNewOrder;

后置条件：创建了一个新的拥有唯一标识的订单实例，初始化该订单实例的属性

②操作：UserLogin(AccountID:String, Password:String);

后置条件：系统创建一个线程管理用户的操作。

③操作：queryMovie(movieId, [cinemaId]);

后置条件：系统查询数据库根据电影 id 返回该电影的详细信息，如果参数含有 cinemaId 则还要返回该电影的上映场次，如果参数不含有 cinemaId 则返回上映该电影的电影院列表。

④操作：queryCinema(cinemaId);

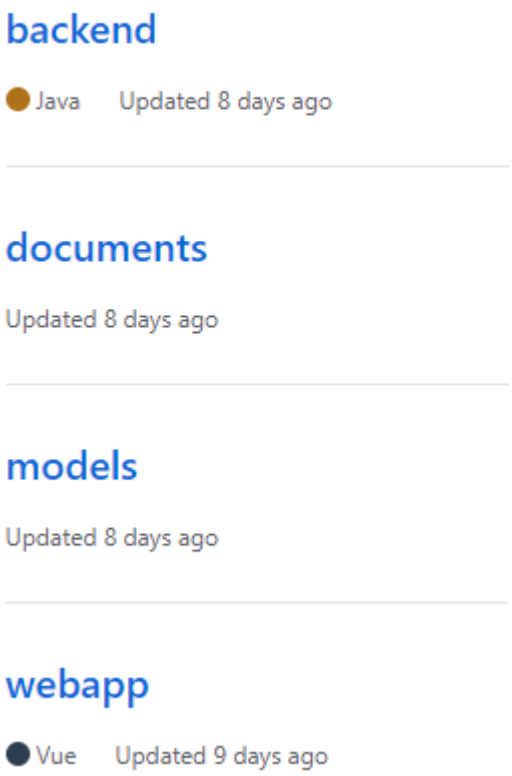
后置条件：系统查询数据库根据电影院 id 返回电影院上映电影列表。

⑤操作：generateOrder(cinemaId, movieId, screenId, seatIds ...);

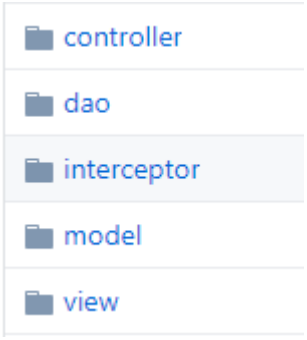
后置条件：系统根据参数信息确定该场次电影有位置，修改数据库锁定该位置后生成订单，往数据库添加订单信息。

三． 模块划分

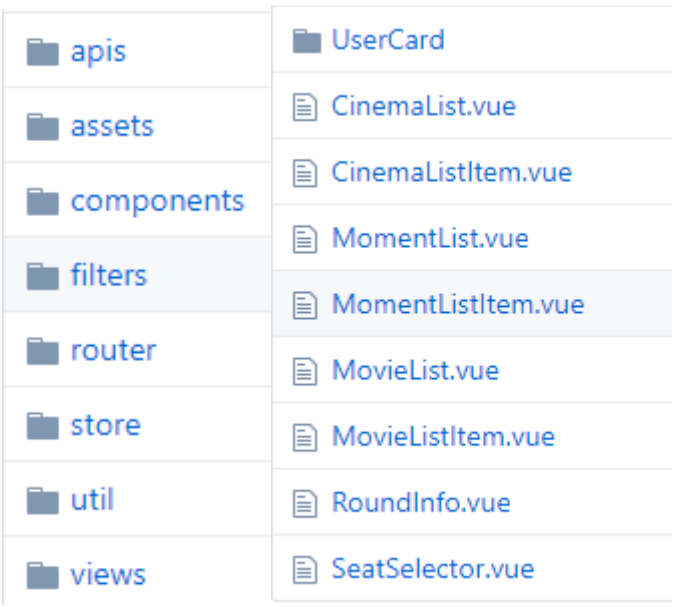
总体模块



后端模块



前端模块



四． 使用的软件设计技术

a. 面向对象的设计

i. 使用 Hibernate 进行 Object Relation Mapping

我们首先假设没有 Hibernate，我们如何自己手动持久化数据呢？方法就是在 DAO 中根据要求生成 SQL。这样做的弊端是：工作量大，容易写错，难以重构，代码充满了字符串操作。Hibernate 主要解决了这个问题，通过 ORM，Java 程序员只需要为 DAO 写签名，Hibernate 自动生成 SQL。例如：

```
1 public interface RoundDao extends JpaRepository<Round, Long> {
2       List<Round> findByMovie_IdAndCinema_Id(long movieId, long cinemaId);
3 }
```

在这个例子中，我们提供了 Entity 类型: Round，主键类型: Long，方法签名，Hibernate 会自动根据它的 Naming Strategy 为你生成对应的函数实现 ,即根据 movieId 和 cinemaId 来查找对应的场次。

当然也可以使用@Query 来自定义 Sql 但一般不建议这样做。如果需要更底层的 API 可以使用 CrudRepository，JpaRepository 是继承自 CrudRepository，具体的不同比如 Jpa 直接返回结果的 List，CrudRepository 返回结果的 Iterator，更适合自定义 DAO，在本次作业我们采用 JpaRepository，因为它提供的功能已经足够强大了。

b. 声明式编程

```
@RestController
public class RoundService {

    @RequestMapping(value = "/api/rounds/{roundId}", method = RequestMethod.GET)
    public Collection<SeatView> getRoundDetail(@PathVariable Long roundId) {
        return watchDao.findById_RoundId(roundId).stream()
                .map(w -> new SeatView(w.getId(), w.getUser().getReportedCount()))
                .collect(Collectors.toList());
    }

    @RequestMapping(value = "/api/rounds/{roundId}/report", method = RequestMethod.POST)
    @Transactional
    public void postReport(@PathVariable Long roundId, @RequestBody ReportDetail[] reportDetails) {
        Arrays.stream(reportDetails).map(r ->
            watchDao.findById_RoundIdAndId_SeatXAndId_SeatY(roundId, r.getSeatX(), r.getSeatY()).stream().map(w ->
                w.getUser().getId()
            ).collect(Collectors.toList())
        ).flatMap(List::stream).forEach(id -> {
            User user = userDao.findOne(id);
            user.incReportedCount();
            userDao.save(user);
        });
    }
}
```

举例说明：

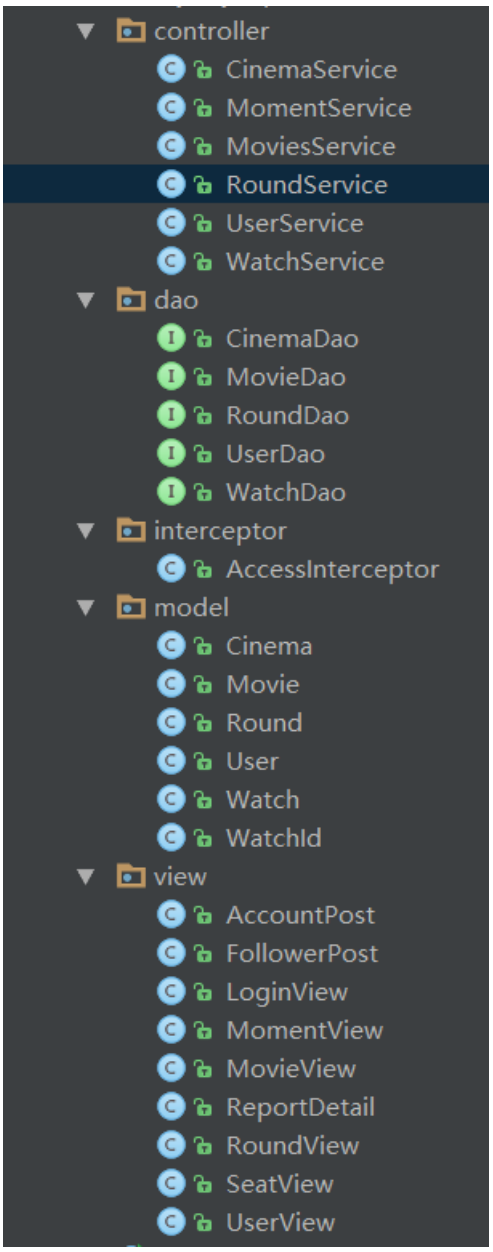
@PathVariable：使用它来提取 url 中的变量

@RequestBody：意思是根据后面参数的类型，对 HTTP Request 进行反序列化

@RequestMapping：控制 http method 和 api

@RestController：相当于@Target, @Retention, @Controller, @ResponseBody

c. MVC 设计


controller
CinemaService
MomentService
MoviesService
RoundService
UserService
WatchService
dao
CinemaDao
MovieDao
RoundDao
UserDao
WatchDao
interceptor
AccessInterceptor
model
Cinema
Movie
Round
User
Watch
WatchId
view
AccountPost
FollowerPost
LoginView
MomentView
MovieView
ReportDetail
RoundView
SeatView
UserView