电影资讯app详细设计规约

[1 引言 3](#_Toc534310622)

[1.1编写目的 3](#_Toc534310623)

[1.2 背景与依据 3](#_Toc534310624)

[1.3 参考资料 3](#_Toc534310625)

[2 系统软件结构 4](#_Toc534310626)

[2.1登录功能的类调用关系图 4](#_Toc534310627)

[2.2注册功能的类调用关系图 4](#_Toc534310628)

[2.3 找片Tab的类调用关系图 5](#_Toc534310629)

[2.4 热映Tab 的类调用关系图 5](#_Toc534310630)

[2.5 “我的信息”Tab类调用关系图 6](#_Toc534310631)

[2.6电影列表类调用关系图 6](#_Toc534310632)

[2.7 电影详情类调用关系图 7](#_Toc534310633)

[2.8 电影关键词搜索类调用关系图 7](#_Toc534310634)

[2.9电影类型搜索类调用关系图 8](#_Toc534310635)

[2.10电影信息统计类调用关系图 8](#_Toc534310636)

[2.11 用户信息统计类调用关系图 9](#_Toc534310637)

[2.12 服务端软件体系结构图 9](#_Toc534310638)

[3 模块设计 10](#_Toc534310639)

[3.1服务端用户注册模块 10](#_Toc534310640)

[3.1.1程序描述 10](#_Toc534310641)

[3.1.2功能 10](#_Toc534310642)

[3.1.3性能 10](#_Toc534310643)

[3.1.4输入项 10](#_Toc534310644)

[3.1.5输出项 10](#_Toc534310645)

[3.1.6.算法 10](#_Toc534310646)

[3.1.7流程图 11](#_Toc534310647)

[3.1.8限制条件及出错处理 11](#_Toc534310648)

[3.2服务端用户登录模块 11](#_Toc534310649)

[3.2.1程序描述 11](#_Toc534310650)

[3.2.2功能 11](#_Toc534310651)

[3.2.3性能 11](#_Toc534310652)

[3.2.4输入项 11](#_Toc534310653)

[3.2.5输出项 11](#_Toc534310654)

[3.2.6.算法 12](#_Toc534310655)

[3.2.7流程图 12](#_Toc534310656)

[3.2.8限制条件及出错处理 12](#_Toc534310657)

[3.3服务端收藏电影模块 12](#_Toc534310658)

[3.3.1程序描述 12](#_Toc534310659)

[3.3.2功能 12](#_Toc534310660)

[3.3.3性能 13](#_Toc534310661)

[3.3.4输入项 13](#_Toc534310662)

[3.3.5输出项 13](#_Toc534310663)

[3.3.6.算法 13](#_Toc534310664)

[3.3.7流程图 14](#_Toc534310665)

[3.3.8限制条件及出错处理 14](#_Toc534310666)

[3.4服务端浏览历史模块 15](#_Toc534310667)

[3.4.1程序描述 15](#_Toc534310668)

[3.4.2功能 15](#_Toc534310669)

[3.4.3性能 15](#_Toc534310670)

[3.4.4输入项 15](#_Toc534310671)

[3.4.5输出项 15](#_Toc534310672)

[3.4.6.算法 15](#_Toc534310673)

[3.4.7流程图 16](#_Toc534310674)

[3.4.8限制条件及出错处理 17](#_Toc534310675)

[3.5服务端数据统计模块 17](#_Toc534310676)

[3.5.1程序描述 17](#_Toc534310677)

[3.5.2功能 17](#_Toc534310678)

[3.5.3性能 17](#_Toc534310679)

[3.5.4输入项 17](#_Toc534310680)

[3.5.5输出项 17](#_Toc534310681)

[3.5.6.算法 17](#_Toc534310682)

[3.5.7流程图 18](#_Toc534310683)

[3.5.8限制条件及出错处理 18](#_Toc534310684)

[3.6服务端电影推荐模块 18](#_Toc534310685)

[3.6.1程序描述 18](#_Toc534310686)

[3.6.2功能 18](#_Toc534310687)

[3.6.3性能 18](#_Toc534310688)

[3.6.4输入项 18](#_Toc534310689)

[3.6.5输出项 19](#_Toc534310690)

[3.6.6.算法 19](#_Toc534310691)

[3.6.7流程图 20](#_Toc534310692)

[3.6.8限制条件及出错处理 20](#_Toc534310693)

[4 类设计说明 20](#_Toc534310694)

[4.1登录注册相关类 20](#_Toc534310695)

[4.1.1程序描述 20](#_Toc534310696)

[4.1.2功能 20](#_Toc534310697)

[4.1.3性能 21](#_Toc534310698)

[4.1.4输入项 21](#_Toc534310699)

[4.1.5输出项 21](#_Toc534310700)

[4.1.6.属性详细描述 21](#_Toc534310701)

[4.1.7方法详细描述及实现流程 21](#_Toc534310702)

[4.1.8类内，类外方法调用状态图 22](#_Toc534310703)

[4.1.9限制条件及出错处理 22](#_Toc534310704)

[4.2电影列表相关类 23](#_Toc534310705)

[4.2.1程序描述 23](#_Toc534310706)

[4.2.2功能 23](#_Toc534310707)

[4.2.3性能 23](#_Toc534310708)

[4.2.4输入项 23](#_Toc534310709)

[4.2.5输出项 23](#_Toc534310710)

[4.2.6.属性详细描述 23](#_Toc534310711)

[4.2.7方法详细描述及实现流程 24](#_Toc534310712)

[4.2.8类内，类外方法调用状态图 24](#_Toc534310713)

[4.2.9限制条件及出错处理 24](#_Toc534310714)

[4.3电影详情相关类 25](#_Toc534310715)

[4.3.1程序描述 25](#_Toc534310716)

[4.3.2功能 25](#_Toc534310717)

[4.3.3性能 25](#_Toc534310718)

[4.3.4输入项 25](#_Toc534310719)

[4.3.5输出项 25](#_Toc534310720)

[4.3.6.属性详细描述 25](#_Toc534310721)

[4.3.7方法详细描述及实现流程 25](#_Toc534310722)

[4.3.8类内，类外方法调用状态图 26](#_Toc534310723)

[4.3.9限制条件及出错处理 27](#_Toc534310724)

## 1 引言

#### 1.1编写目的

a. 对电影资讯app概要设计规约进行完善。

b. 详细说明系统各个类的具体实现方式与流程。

#### 1.2 背景与依据

a. 主要的功能实现的需求依据，详见电影资讯app需求分析规约。

b. 参考了google官方mvp架构示例项目-todoapp。

#### 1.3 参考资料

a. 《软件工程-实践者的研究方法（原书第八版）》-罗杰S·普莱斯曼，机械工业出版社

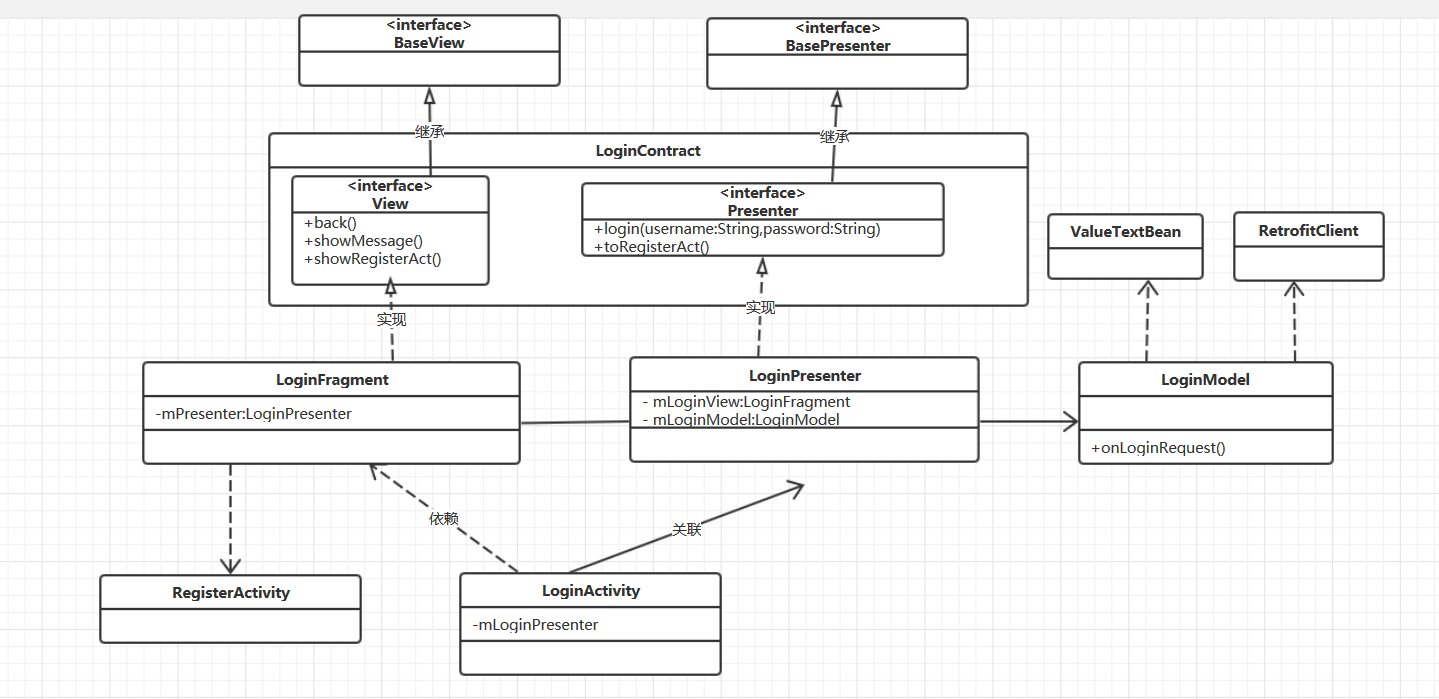
b. 《第一行代码-Android》人民邮电出版社

c. 《基于本体用户兴趣模型的个性化推荐算法》-[严隽薇](http://xueshu.baidu.com/s?wd=author%3A%28%E4%B8%A5%E9%9A%BD%E8%96%87%29%20%E5%90%8C%E6%B5%8E%E5%A4%A7%E5%AD%A6CIMS%E7%A0%94%E7%A9%B6%E4%B8%AD%E5%BF%83&tn=SE_baiduxueshu_c1gjeupa&ie=utf-8&sc_f_para=sc_hilight%3Dperson)，[黄勋](http://xueshu.baidu.com/s?wd=author%3A%28%E9%BB%84%E5%8B%8B%29%20%E5%B9%BF%E8%A5%BF%E6%B0%91%E6%97%8F%E5%A4%A7%E5%AD%A6%E7%9B%B8%E6%80%9D%E6%B9%96%E5%AD%A6%E9%99%A2&tn=SE_baiduxueshu_c1gjeupa&ie=utf-8&sc_f_para=sc_hilight%3Dperson)，[刘敏](http://xueshu.baidu.com/s?wd=author%3A%28%E5%88%98%E6%95%8F%29%20%E5%90%8C%E6%B5%8E%E5%A4%A7%E5%AD%A6CIMS%E7%A0%94%E7%A9%B6%E4%B8%AD%E5%BF%83&tn=SE_baiduxueshu_c1gjeupa&ie=utf-8&sc_f_para=sc_hilight%3Dperson).

d. 《基于协同过滤算法的电影个性化推荐系统设计与实现》-贾忠涛

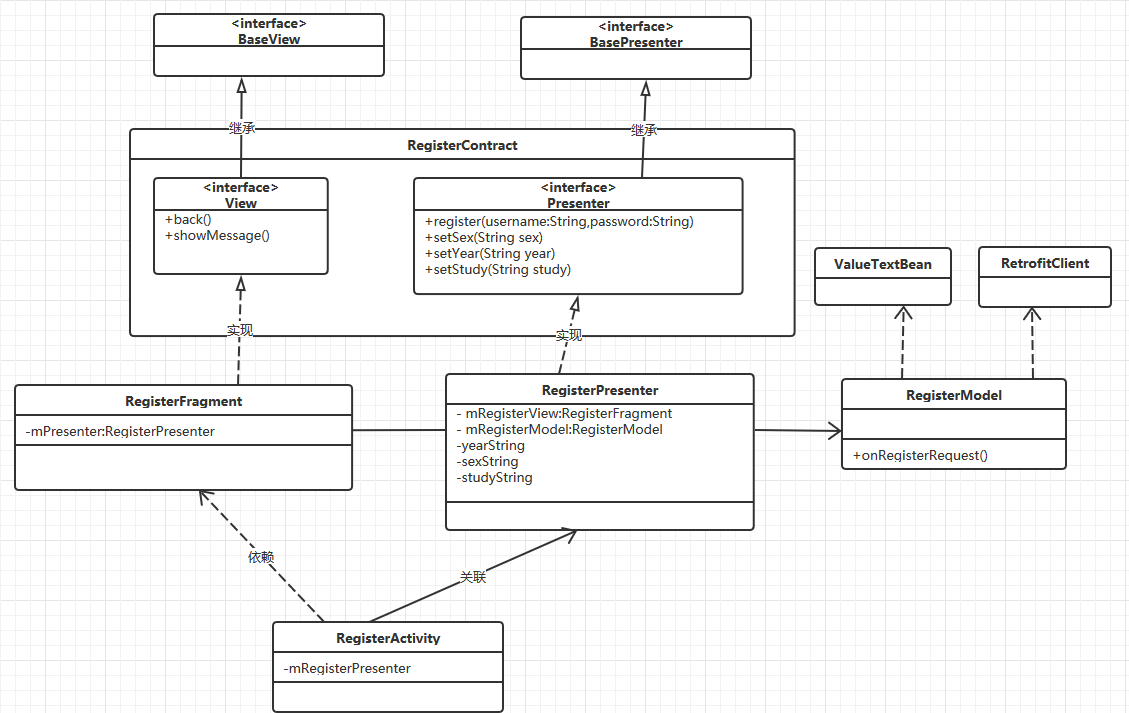
## 2 系统软件结构

#### 2.1登录功能的类调用关系图



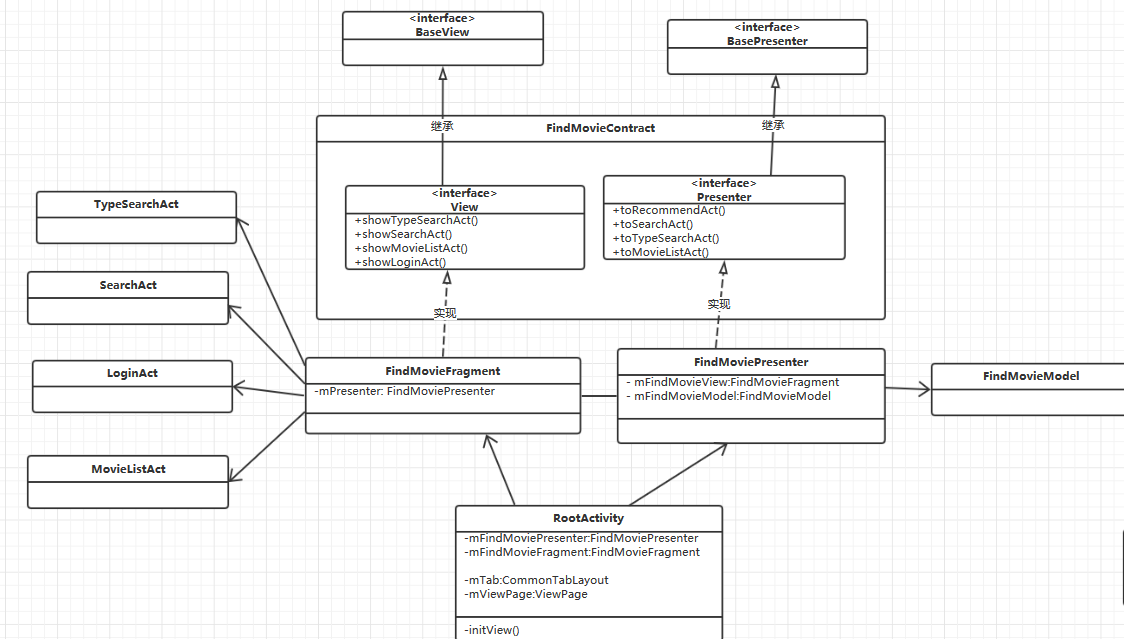
本系统中的登录功能，主要是在LoginActivity中进行MVP三部分的初始化与关系组织，当用户输入相应信息后点击登录饥按钮，会调用Presenter的login函数，对用户名和密码基本格式进行判断（包括是否为空，长度过长等），接着调用Model的onLoginRequest函数，生成RetrofitClient访问服务器，用ValueTextBean接受登录信息，再由presenter调用view来显示对应信息。

#### 2.2注册功能的类调用关系图



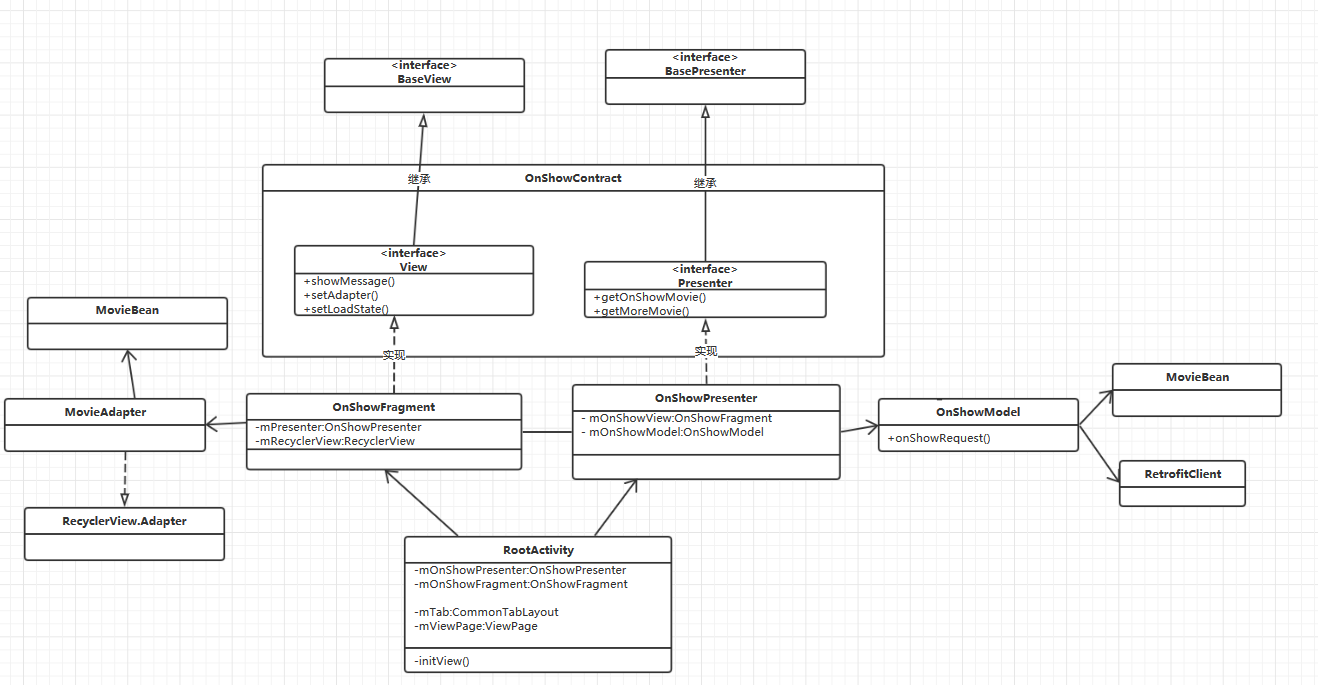
本系统的注册功能，是在RegisterActivity中进行MVP三部分的初始化与关系组织，由用户对View上的下拉框控件进行操作，选择用户的性别，年龄，学历等，在输入用户名密码后，再调用Presenter函数中的register函数进行注册，register中调用了Model的onRegisterRequest函数访问后端服务器，完成用户信息表的插入。

#### 2.3 找片Tab的类调用关系图



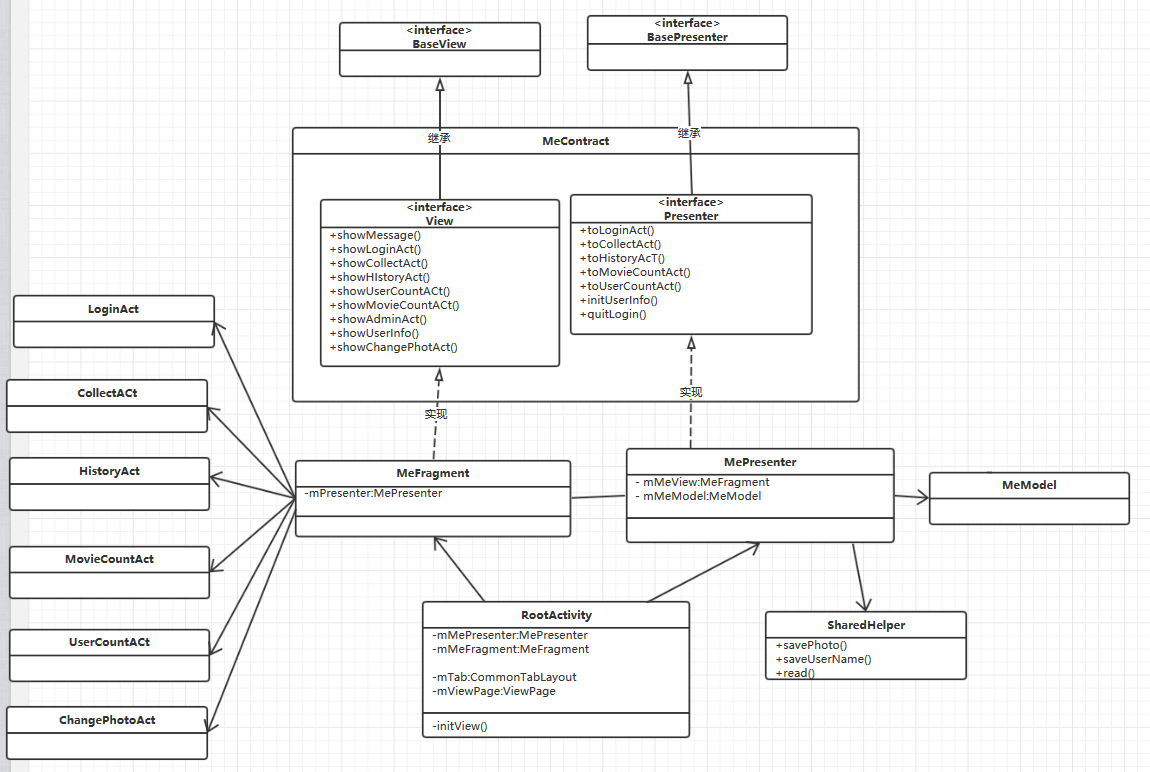
这是打开APP首先展示的View, RootActivity的第一个Tab项，主要是一些导航性质的功能，能够跳转到其他功能的Activity中。包括LoginActivity, MovieListActivity, SearchActivity, TypeSearchActivity。

#### 2.4 热映Tab 的类调用关系图



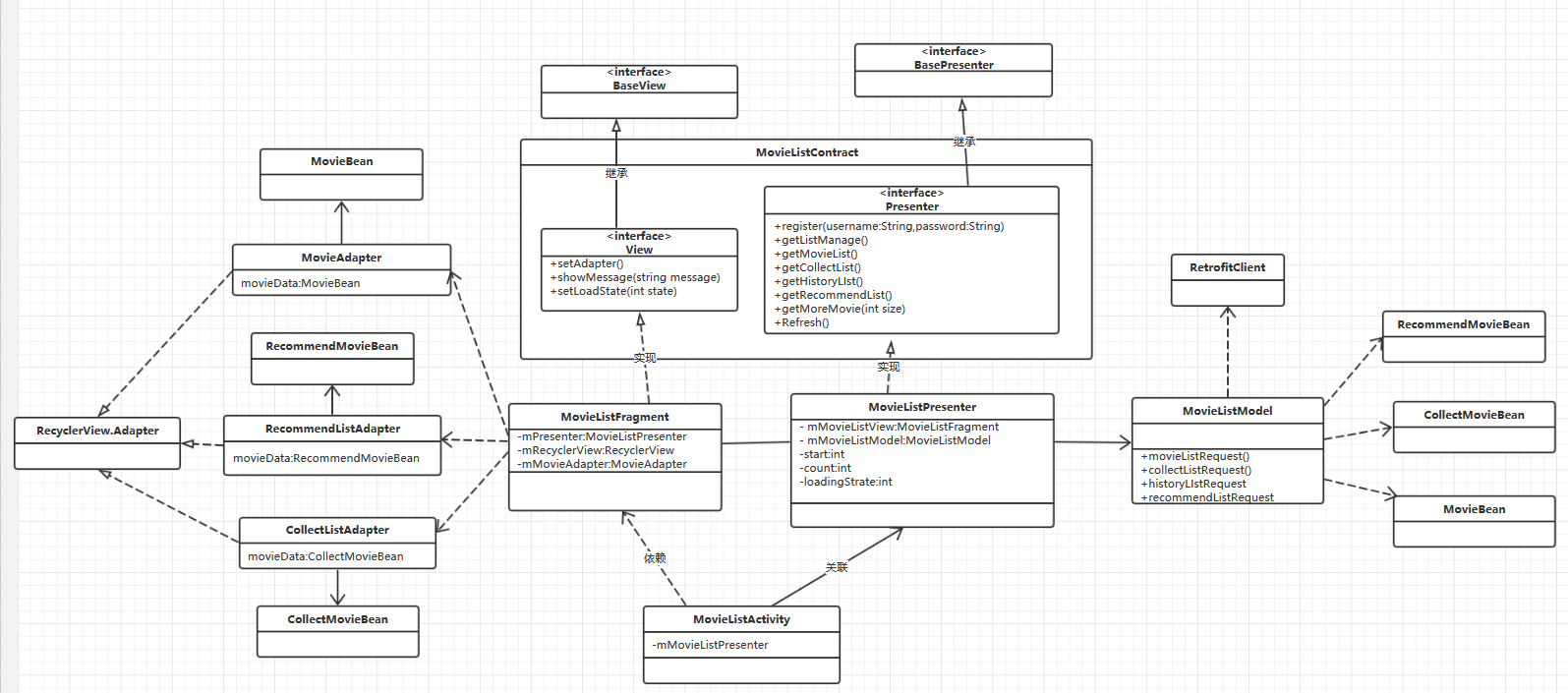
这是RootActivity的第二个Tab,主要是通过Presenter的getOnShowMovie函数，调用OnshowModel的onShowRequest函数访问豆瓣API,获取电影信息，返回后转化成MovieBean，由MovieAdapter类绑定到视图上。

#### 2.5 “我的信息”Tab类调用关系图



这是RootActivity的第三个Tab界面，主要也是一些导航性质的功能，跳转到登录界面，查看收藏界面，查看浏览记录界面，查看电影数据统计界面，查看用户数据统计界面，更换头像界面。还有一个退出登录的功能，通过sharedHelper类来对SharePrefence进行读写，更改本地保存的用户信息。

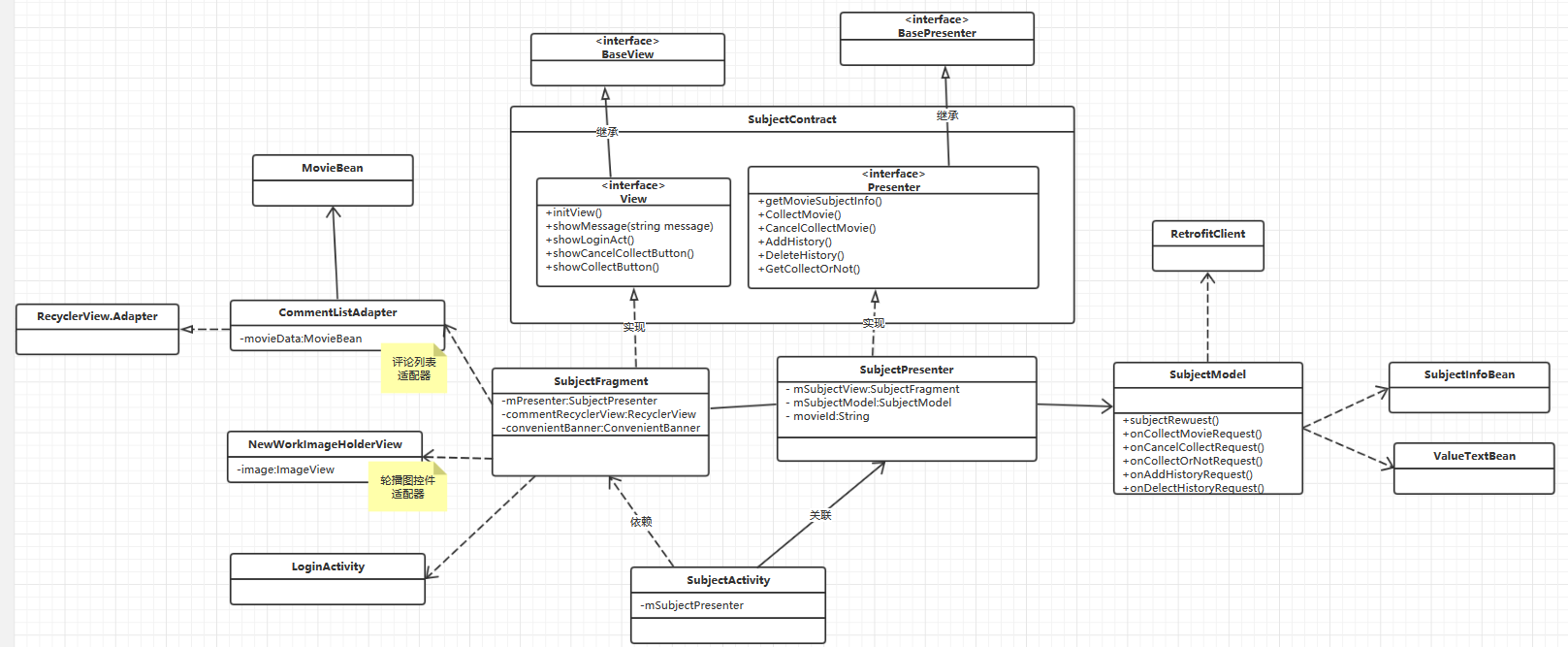
#### 2.6电影列表类调用关系图



电影列表类，主要功能是负责获取并展示各种列表信息，包括电影列表，收藏列表，推荐列表，历史记录列表。其中电影列表包括热映电影列表，关键词搜索结果的电影列表，新片榜电影列表等等。

主要流程是通过Presenter的getListManage()函数，判断是需要生产何种类型的列表，然后通过相关的getXXXList函数调用MovieListModel中的XXXListRequest来访问服务器获取列表信息。获得到的信息，通过Fragment中的Adapter类来对数据进行绑定，展示在RecycleView即列表控件上。

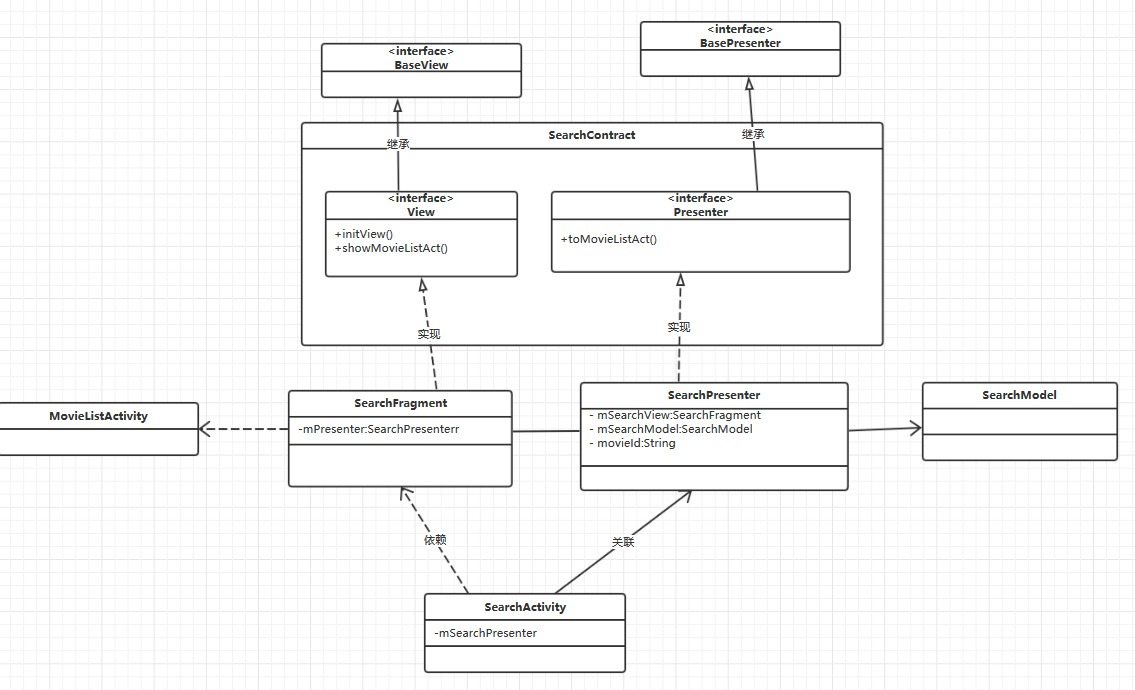
#### 2.7 电影详情类调用关系图



电影详情类，主要是在用户点击电影列表中某个项后进入电影详情界面，对该电影的详细信息进行展示，增加用户浏览记录，提供收藏电影，取消收藏，删除浏览记录的功能。如果用户未登录时进行收藏电影等操作，会跳转到登录界面。

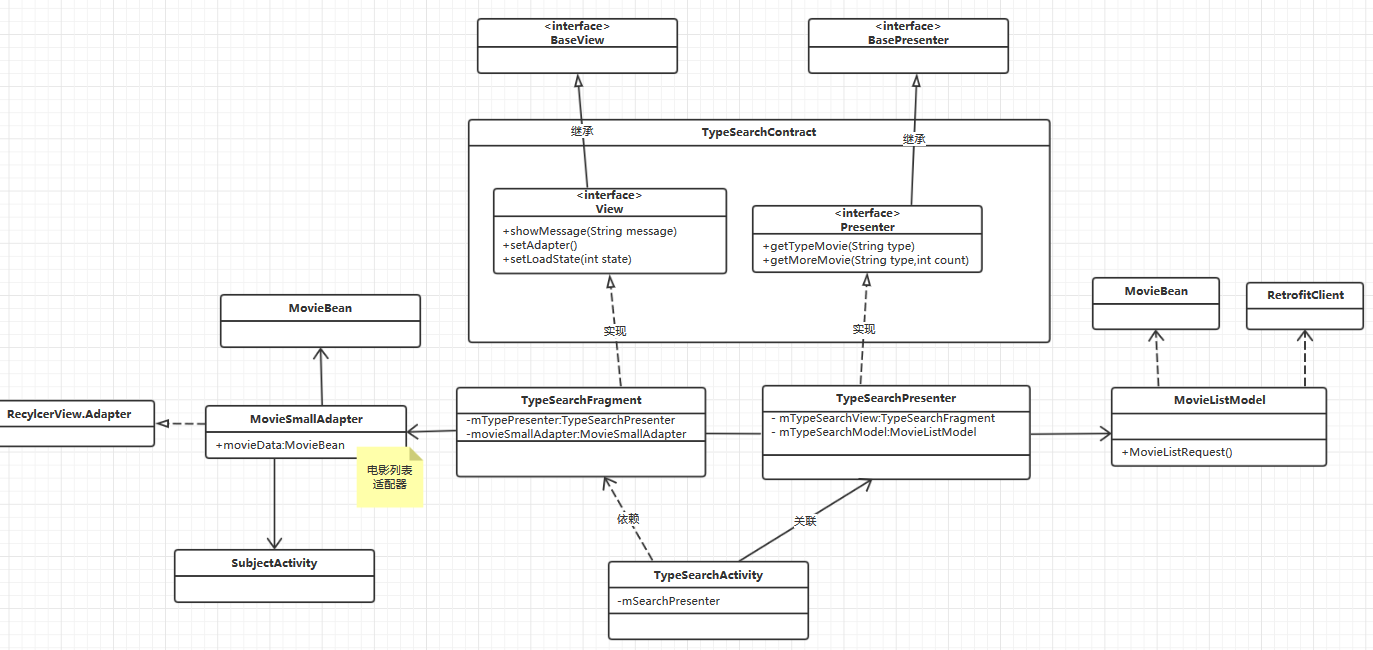
Fragment中initView函数用于初始化界面信息，showLoginAct用于跳转到登录界面，showCancelCollectButton与showCollectButton根据用户的收藏与否情况，对收藏按钮进行设置。Presenter中getMovieSubjecInfo()对应model的subjectRequest(),用于访问豆瓣API获取电影详情信息，collectMovie()用于收藏电影，cancelCollect()取消收藏，addHistory()增加浏览记录，deleteHistory()删除浏览记录。

#### 2.8 电影关键词搜索类调用关系图



这是一个比较简单的类，引用了一个外部开源库的SearchView控件，会对用户的搜索历史进行保存并且持久性存储在本地数据库中。在用户输入关键词后，会跳转到MovieListActivity获取关键词的搜索结果，并对搜索结果进行展示。

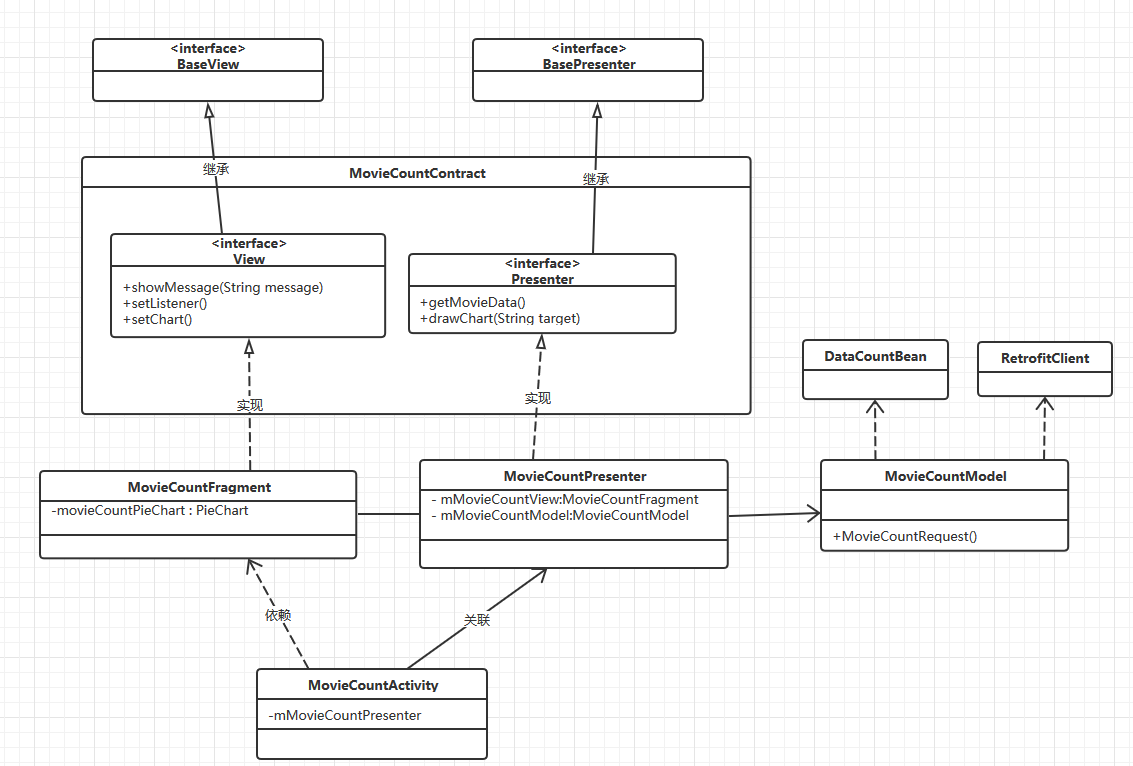
#### 2.9电影类型搜索类调用关系图



电影分类搜索类，主要功能是调用豆瓣API获得各类型电影数据，在使用RecyclerView进行展示。

其中使用presenter来调用MovieListModel的MovieListRequest()来访问豆瓣API,返回的数据存在MovieBean类中，交由MovieSmallAdapter进行数据绑定，并且点击各个项会跳转到对应的电影详情界面。

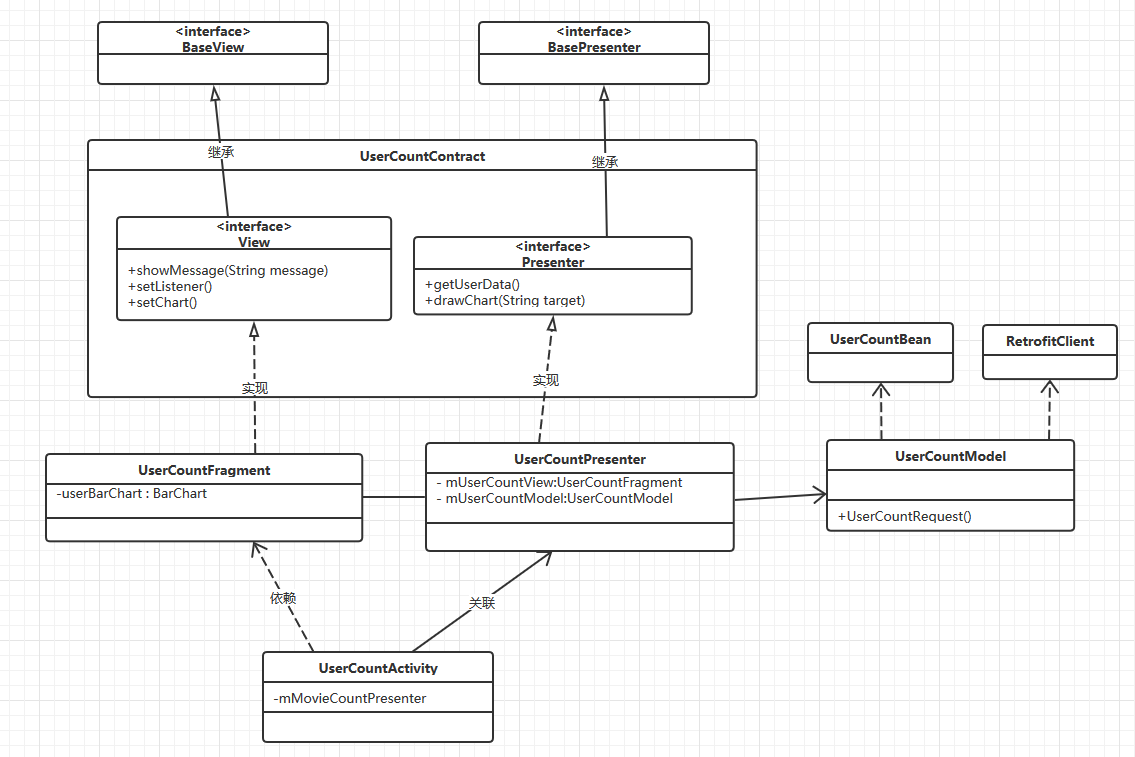
#### 2.10电影信息统计类调用关系图



电影信息统计类，能够根据用户所选的指标，访问服务器获得对应电影统计数据，以图形化的形式展示给用户。

主要是通过Presenter的getMovieData函数，调用Model的MovieCountRequest方法，访问服务器获得数据存再DataCountBean中，通过Fragment的setChart函数完成图表的绘制。

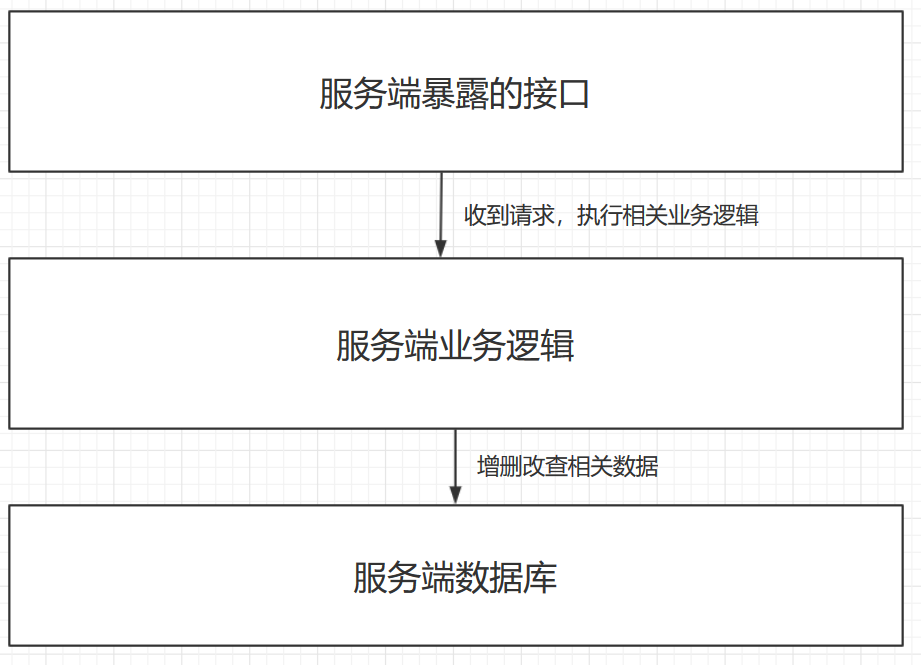
#### 2.11 用户信息统计类调用关系图



用户信息统计类，能够根据用户所选的指标，访问服务器获得对应用户信息统计数据，以图形化的形式展示给用户。

主要是通过Presenter的getUserData函数，调用Model的UserCountRequest方法，访问服务器获得数据存再UserCountBean中，通过Fragment的setChart函数完成图表的绘制。

#### 2.12 服务端软件体系结构图



服务端主要是采用node.js 下的express框架，为客户端提供了功能接口，具体接口信息详见概要设计规约2.3.6接口设计。

服务端的业务逻辑主要是对数据库进行相应的增删改查，向客户端返回对应数据。较为复杂的逻辑主要是用户喜好特征表的计算，根据基于内容的推荐算法，与基于用户画像的协同过滤推荐算法来给用户推荐他可能喜欢的电影。

服务端数据库使用了MongoDB数据库来进行数据的存储与访问。

## 3 模块设计

#### 3.1服务端用户注册模块

###### 3.1.1程序描述

响应客户端注册请求，检查注册数据，正确无误则向数据库插入一条新用户信息，失败则返回失败信息。

###### 3.1.2功能

为客户端的注册请求提供接口，完成注册功能，即在数据库中插入一条新的用户记录。

###### 3.1.3性能

应该在十秒内向客户端返回信息。

###### 3.1.4输入项

用户名 req.query.username ，密码 req.query.password 用户性别 req.query.sex

用户年龄 req.query.year 用户学历 req.query.study 用户头像 req.query.photo

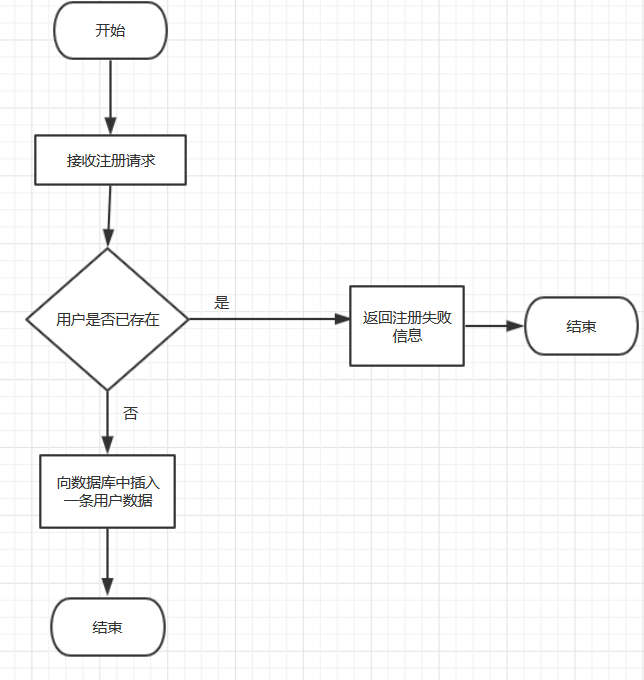
###### 3.1.5输出项

返回json数据 {value：string, Text : string}, 其中value表示成功或失败，Text表示成功信息或失败信息。

###### 3.1.6.算法

先根据用户查找用户表，判断该用户名是否已经被注册，若已经被注册则返回错误信息，若没有，则向数据库中插入一条记录，返回注册成功信息。

###### 3.1.7流程图



###### 3.1.8限制条件及出错处理

用户名不能与已有用户名重复，出错则返回注册失败信息以及原因

#### 3.2服务端用户登录模块

###### 3.2.1程序描述

响应客户端登录请求，检查用户名与密码是否正确，正确无误则返回登录成功信息，有错误则返回失败信息。

###### 3.2.2功能

为客户端的登录请求提供接口，完成登录功能，即在数据库中查询相关的用户记录，并进行判断。

###### 3.2.3性能

应该在十秒内向客户端返回信息。

###### 3.2.4输入项

用户名 req.query.username ，密码 req.query.password

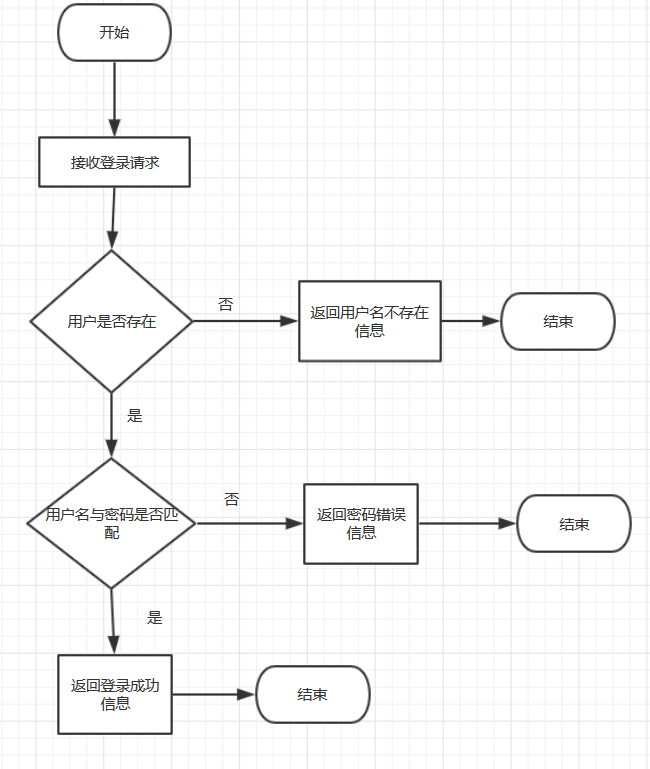
###### 3.2.5输出项

返回json数据 {value：string, Text : string}, 其中value表示成功或失败，Text表示成功信息或失败信息。

###### 3.2.6.算法

先根据用户名查找用户表，判断该用户名是否存在，若不存在则返回用户名错误信息，若存在则比对用户输入的密码与数据库中的密码，若匹配则返回登录成功信息，失败则返回密码错误信息。

###### 3.2.7流程图



###### 3.2.8限制条件及出错处理

用户名与密码需正确匹配，出错则返回报错信息。

#### 3.3服务端收藏电影模块

###### 3.3.1程序描述

响应客户端收藏功能请求，包括用户收藏电影，用户取消收藏，判断某电影收藏与否，收藏列表展示。

###### 3.3.2功能

为客户端的收藏功能请求提供接口，完成用户收藏电影，在收藏记录表中插入一条记录；用户取消收藏即删除一条记录，用户收藏列表展示即查找该用户的所有收藏记录，判断某电影收藏与否即查找是否存在该用户收藏该电影的记录。

###### 3.3.3性能

应该在十秒内向客户端返回信息。

###### 3.3.4输入项

收藏电影：用户名 username: string, 电影ID movieId: string, 电影名movieTitle: string，电影类型 movieType: string，电影导演 movieDirector: string，电影年代 movieYear : string，电影语言 movieLanguage: string，用户收藏时间time:string.

取消收藏：用户名 username: string, 电影ID movieId: string

收藏列表展示：用户名 username: string,

判断收藏与否：用户名 username: string, 电影ID movieId: string

###### 3.3.5输出项

收藏电影，取消收藏，判断收藏与否：返回json数据 {value：string, Text : string}, 其中value表示成功或失败，Text表示成功信息或失败信息。

收藏列表展示：{subject: Array[{用户名username: string, 电影ID movieID: string , 电影名 movieTitle: string , 收藏时间 time:string}]}

###### 3.3.6.算法

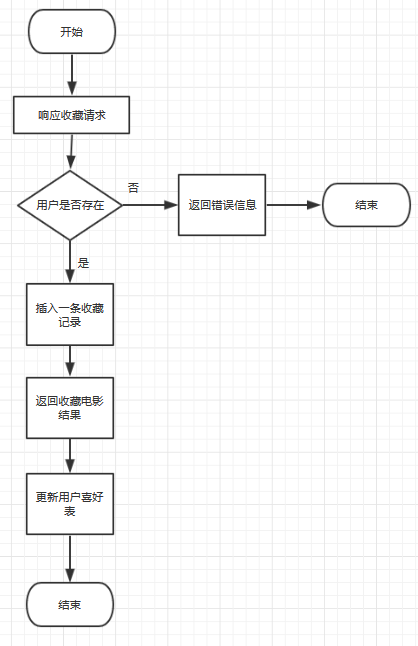
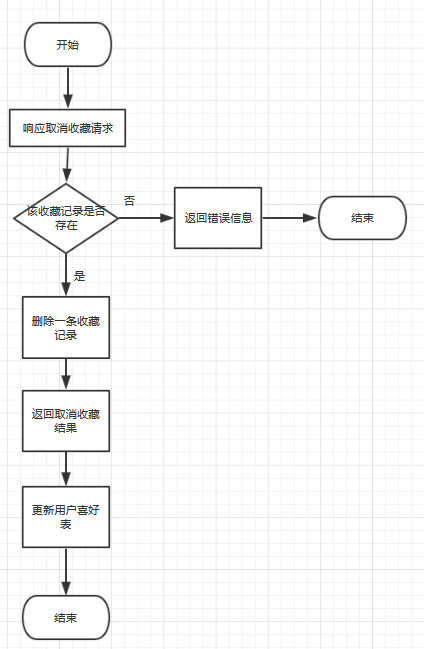
收藏电影：判断该用户是否存在，存在则在用户收藏表中插入一条用户收藏记录，否则返回报错信息。

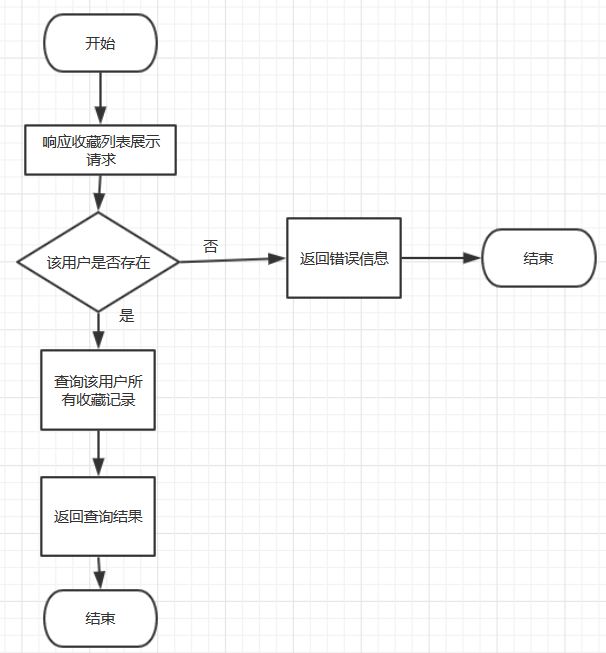
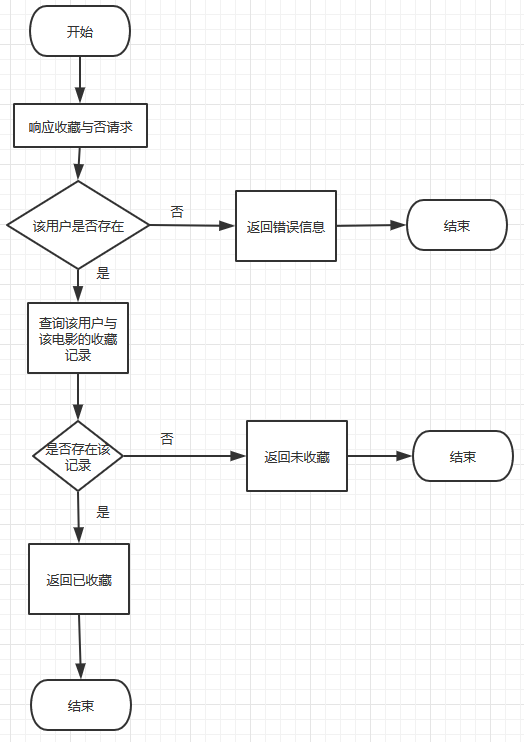
取消收藏：判断该收藏记录是否存在，存在则在用户收藏表中删除一条用户记录，否则返回报错信息。

用户收藏列表展示：判断该用户是否存在，存在则在用户收藏表中查找某用户的所有记录，转换成json格式返回，不存在则返回报错信息

判断收藏与否：判断该用户是否存在，不存在则返回报错信息，存在则在用户表中查找某用户收藏某电影的记录，找到返回该电影已收藏，没找到则返回该电影未收藏。

###### 3.3.7流程图

收藏电影：取消收藏:

收藏列表展示：收藏与否

###### 3.3.8限制条件及出错处理

收藏电影，收藏列表展示，判断收藏与否：用户名需已注册，否则会返回报错信息。

取消收藏：该收藏记录需存在，否则会返回报错信息。

#### 3.4服务端浏览历史模块

###### 3.4.1程序描述

响应客户端收藏功能请求，包括添加浏览记录，删除浏览记录，浏览记录列表展示。

###### 3.4.2功能

为客户端的浏览记录功能请求提供接口，用户添加浏览记录，在收浏览记录表中插入一条记录；用户删除浏览记录即删除表中一条记录，用户浏览记录列表展示即查找该用户的所有浏览记录。

###### 3.4.3性能

应该在十秒内向客户端返回信息。

###### 3.4.4输入项

增加浏览记录：用户名 username: string, 电影ID movieId: string, 电影名movieTitle: string，用户浏览时间time:string.

删除浏览记录：用户名 username: string, 电影ID movieId: string

浏览记录列表展示：用户名 username: string,

###### 3.4.5输出项

增加浏览记录，删除浏览记录：返回json数据 { value：string, Text : string } , 其中value表示成功或失败，Text表示成功信息或失败信息。

浏览记录列表展示：{subject: Array[{用户名username: string, 电影ID movieID: string , 电影名 movieTitle: string , 收藏时间 time:string}]}

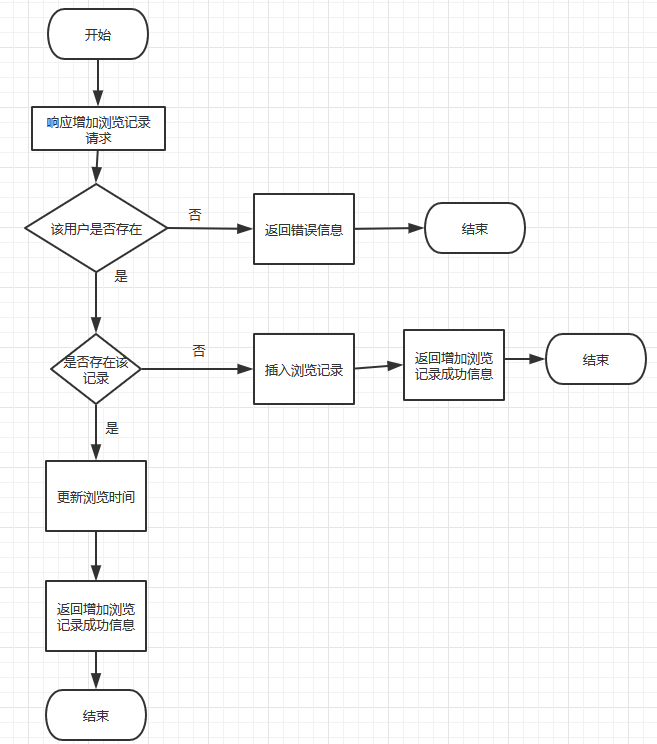
###### 3.4.6.算法

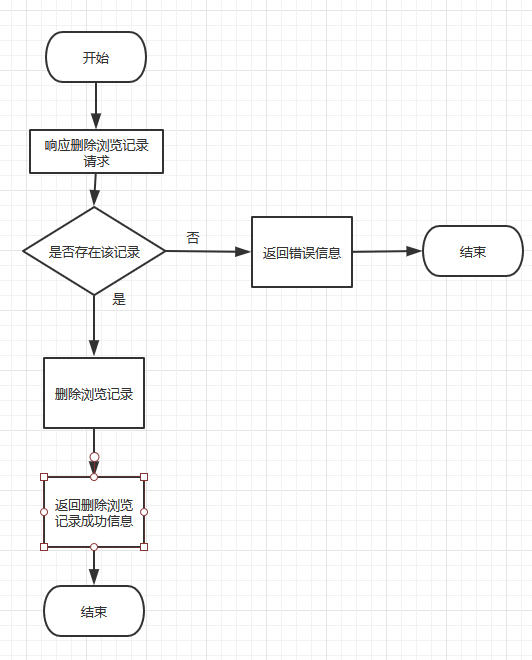
增加浏览记录：判断该用户是否存在，存在则接着判断浏览记录是否存在，不存在就插入一条浏览记录表，浏览记录已存在就更新浏览时间，否则返回报错信息。

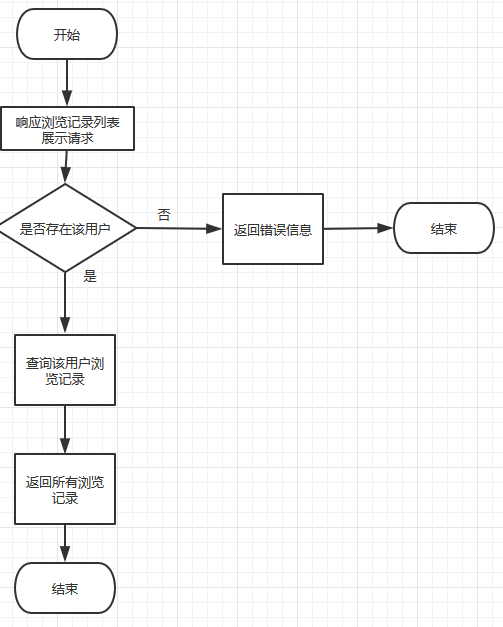
取消收藏：判断该记录是否存在，存在则在用户浏览记录表中删除一条用户记录，否则返回报错信息。

用户浏览记录列表展示：判断该用户是否存在，存在则在用户浏览表中查找某用户的所有记录，转换成json格式返回。否则返回报错信息。

###### 3.4.7流程图

增加浏览记录：

删除浏览记录：

浏览记录列表：

###### 3.4.8限制条件及出错处理

增加浏览记录，浏览记录列表展示：用户名需已注册，否则会返回报错信息。

删除浏览记录：该浏览记录需存在，否则会返回报错信息。

#### 3.5服务端数据统计模块

###### 3.5.1程序描述

响应客户端数据统计请求，统计数据库中所有的用户信息，主要数据包括各类用户的数量，统计数据库中所有电影收藏记录信息，主要数据包括各类型电影收藏比例，各年代电影收藏比例，各语种电影收藏比例。

###### 3.5.2功能

为客户端的数据统计请求提供接口，完成数据统计功能，即在数据库中查询相关的用户记录与电影记录，并进行统计。

###### 3.5.3性能

应该在十秒内向客户端返回信息。

###### 3.5.4输入项

无。

###### 3.5.5输出项

电影数据统计{TypeLike: Array[{类型名 name:string，该类型电影收藏百分比 count:float}],

LanguageLike:Array[{语种名 name:string，该语种电影收藏百分比 count:float}]

YearLike:Array[{年代名 name:string，该年代电影收藏百分比 count:float}]}

用户数据统计{manYear: Array[{年龄段名 name:string，该年龄段男性用户人数 count:float}],

womanYear: Array[{年龄段名 name:string，该年龄段女性用户人数 count:float}],

manStudy: Array[{学历名 name:string，该学历段男性用户人数 count:float}],

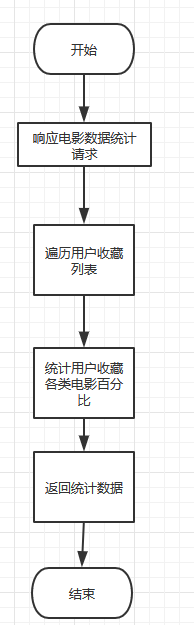
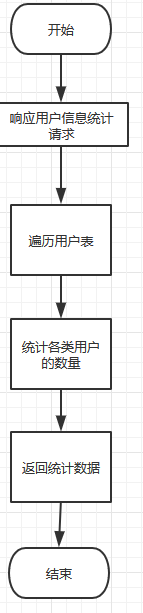
womanYear: Array[{学历名 name:string，该学历段女性用户人数 count:float}],}

###### 3.5.6.算法

电影数据统计：查询电影收藏列表，对每一项的年代，类型，语种进行统计，将各个类型，各语种，各年代电影的百分比通过json格式返回给客户端。

用户数据统计：查询用户列表，对每一个用户的年龄段，性别，学历进行统计，将各个年龄段的男女用户，各个学历的男女用户数量通过json格式返回给客户端。

###### 3.5.7流程图

电影收藏数据统计 用户信息数据统计

###### 3.5.8限制条件及出错处理

无

#### 3.6服务端电影推荐模块

###### 3.6.1程序描述

响应客户端电影推荐请求，根据用户的收藏信息，以及用户的年龄性别学历等信息来对用户进行电影推荐。

###### 3.6.2功能

为客户端的电影推荐请求提供接口，完成电影推荐功能，即根据特定的电影推荐算法在待选电影数据库中选取用户最可能喜欢的电影来推荐给用户。

###### 3.6.3性能

应该在十秒内向客户端返回信息。

###### 3.6.4输入项

用户名 req.query.username

###### 3.6.5输出项

推荐电影列表：{subject: Array[{电影ID movieId: string, 电影名movieTitle: string，电影类型 movieType: string，电影导演 movieDirector: string，电影年代 movieYear : string，电影语言 movieLanguage: string, 推荐理由reason:string }]}

###### 3.6.6.算法

基于内容的推荐算法（Content-Based Recommendations CB）:

CB的过程一般包括以下三步：

物品表示（Item Representation）：为每个电影抽取出一些特征（也就是电影的年代，类型，语种等）来表示此电影；

特征学习（Profile Learning）：利用一个用户过去喜欢（及不喜欢）的电影的特征数据，来学习出此用户的喜好特征（即用户喜好表）；

生成推荐列表（Recommendation Generation）：通过比较上一步得到的用户喜好表与候选电影的特征，为此用户推荐一组相关性最大的电影。

基于协同过滤的用户画像推荐算法，包括以下三个步骤:

确定用户的属性：年龄段，性别，学历

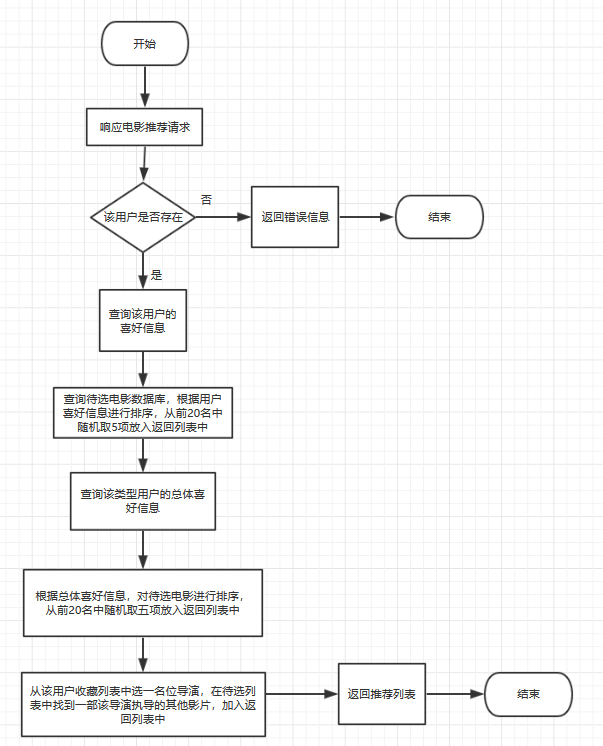
根据所有用户的收藏情况，生成用户画像（即各个性别，年龄段，学历的用户会喜欢什么样的电影），存在typicalUser用户画像表中。

生成推荐列表：通过用户画像中某类用户的电影喜好表来给用户推荐电影。

基于导演的电影推荐：

这是一个比较简单但实用的电影推荐算法，向用户推荐他收藏列表中的导演执导的其他电影。

###### 3.6.7流程图



###### 3.6.8限制条件及出错处理

用户名需要已经注册，否则会返回报错信息；用户应至少收藏了一部分电影，推荐算法的有效性才会比较可观。

## 4 类设计说明

#### 4.1登录注册相关类

###### 4.1.1程序描述

包括LoginActivity, LoginFragment, LoginPresenter, LoginModel, RegisterActivity, RegisterFragment, RegisterPresenter, RegisterModel等类，通过Actitvity进行Model，View, Presenter的绑定，通过model进行数据的获取，通过Presenter来进行登录注册业务逻辑的处理，通过View来进行数据的展示。

###### 4.1.2功能

在客户端提供登录注册界面，向服务端发送登录注册请求，修改当前用户登录状态。

###### 4.1.3性能

###### 4.1.4输入项

通过EditText 输入用户名，密码。

通过Spinner下拉框选择用户的性别，年龄，学历

###### 4.1.5输出项

返回登录成功与否信息，返回注册成功与否信息。

###### 4.1.6.属性详细描述

LoginPresenter类属性：

LoginFragment mLoginFragmen// 登录View

LoginModel mLoginModel //登录model

LoginFragment类属性：

LoginPresenter mPresenter//登录presenter

RegisterPresenter类属性：

RegisterFragment mRegisterFragmen//注册View

RegisterModel mRegisterModel //注册model

String userYear //用户年龄

String userSex //用户性别

String userStudy //用户学历

RegisterFragment类属性：

RegisterPresenter mPresenter//注册presenter

###### 4.1.7方法详细描述及实现流程

LoginPresenter类方法：

void login(String username,String password);//调用LoginModel发送登录请求

void toRegisterAct();//前往注册界面，调用LoginView的showRegisterAct函数

LoginFragment类方法：

void showMessage(String message);//显示信息  
void back();//返回上一个界面

void showRegisterAct();//展示注册界面

LoginModel类方法：

onLoginRequest（String username, String password）//访问服务端登录接口

RegisterPresenter类方法：

void register(String username,String password);//调用RegisterModel发送登录请求

void setYear(String year);//获取用户年龄

void setSex(String sex);//获取用户性别

void setStudy(String study);//获取用户学历

RegisterFragment类方法：

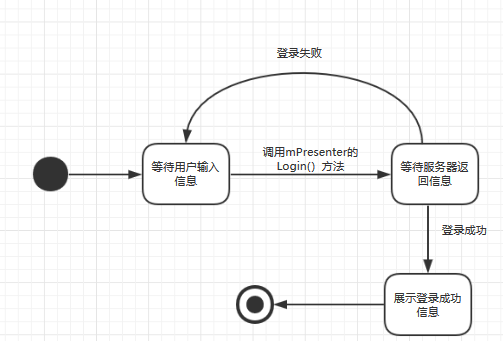
void showMessage(String message);//显示信息  
void back();//返回上一个界面

RegisterModel类方法：

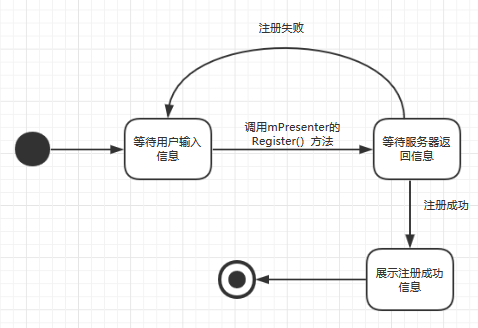
onRegisterRequest（Map<String,String> map）//访问服务端注册接口，参数存储在map中

###### 4.1.8类内，类外方法调用状态图

LoginView类调用状态图：



RegisterView类调用状态图：



###### 4.1.9限制条件及出错处理

用户名与密码不能为空，长度不能超过15个字符。

#### 4.2电影列表相关类

###### 4.2.1程序描述

包括MovieListActivity, MovieListFragment, MovieListPresenter, MovieListModel,等类，通过Actitvity进行Model，View, Presenter的绑定，通过model进行数据的获取，通过Presenter来进行列表与数据的绑定处理，通过View来进行数据的展示。

###### 4.2.2功能

在客户端展示电影信息列表，访问外部豆瓣API获取电影信息，访问服务端获得收藏列表，历史记录列表与推荐电影列表。

###### 4.2.3性能

###### 4.2.4输入项

列表的类型。

###### 4.2.5输出项

在View上输出显示列表信息。

###### 4.2.6.属性详细描述

MovieListPresenter类属性：

MovieListFragment mMovieListFragmen// 电影列表View

MovieListModel mMovieListModel //电影列表model

private int totalCount;//电影列表总长度，用于分页  
private int start = 0;//当前电影列表长度，用于分页  
private int count = 20;//单次获取列表项数目，用于分页  
private MovieBean movieData;//电影信息数据  
private int loading=0;//加载状态，0位未在加载，1位正在加载电影

MovieListFragment类属性：

MovieListPresenter mPresenter//电影列表presenter

RecyclerView mRecyclerView;//列表控件  
MovieAdapter mMovieAdapter;//列表控件适配器

###### 4.2.7方法详细描述及实现流程

MovieListPresenter类方法：

void getMovieList(Intent intent);//获取电影列表信息

void getListManage(Intent intent);//选择生成何种列表信息

void getCollectList();//获取收藏列表

void getHistoryList();//获取历史记录列表

void getRecommendList();//获取推荐列表

void getMoreMovie(int size,Intent intent);//加载更多电影

void afterBackRefresh();//返回时刷新

MovieListFragment类方法：

void showMessage(String message);//展示信息

void setAdapter(MovieBean dataBean);//设置电影列表适配器

void setAdapter(CollectMovieBean dataBean);//设置收藏列表适配器

void setAdapter(RecommendMovieBean dataBean);//设置推荐列表适配器

void setLoadState(int state);

MovieListModel类方法：

public void movieListRequest(String path,Map map)//获取电影列表请求

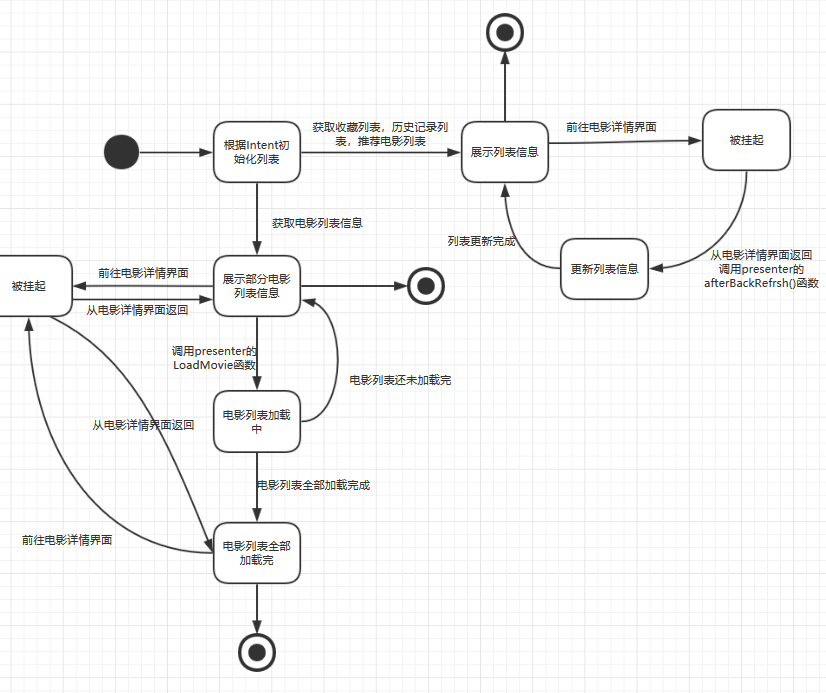
public void collectListRequest(String username)//获取收藏列表请求

public void historyListRequest(String username)//获取历史记录列表请求

public void recommendListRequest(String username)//获取推荐电影列表请求

###### 4.2.8类内，类外方法调用状态图

MovielistView类调用状态图：



###### 4.2.9限制条件及出错处理

网络数据加载应该在10s之内完成，否则会被认定为连接超时放弃连接。如果数据加载失败，可以退出activity后重新加载

#### 4.3电影详情相关类

###### 4.3.1程序描述

包括SubjectActivity, SubjectFragment, SubjectPresenter, SubjectModel,等类，通过Actitvity进行Model，View, Presenter的绑定，通过model进行电影详情数据的获取，通过Presenter来进行收藏，取消收藏，添加，删除历史记录逻辑的处理，通过View来进行电影详情数据的展示。

###### 4.3.2功能

在客户端展示电影详情信息，访问外部豆瓣API获取电影信息，访问服务端获得电影收藏信息，进行收藏电影与取消收藏功能，访问服务端添加或删除浏览记录。

###### 4.3.3性能

###### 4.3.4输入项

要展示的movieID

###### 4.3.5输出项

在View上输出显示电影详细信息。

###### 4.3.6.属性详细描述

SubjectPresenter类属性：

SubjectFragment mSubjectFragment//电影详情view

SubjectModel mSunjectModel//电影详情model

String movieID//电影ID;

MovieSubjectBean dataBean//电影信息数据

SubjectFragment类属性：

SubjectPresenter mPresenter//电影详情presenter

RecyclerView commentRecyclerView;//评论列表控件

ConvenientBanner convenientBanner;//顶部广告栏控件

###### 4.3.7方法详细描述及实现流程

SubjectPresenter类方法：

void getMovieSubjectInfo();//调用SubjectModel,访问豆瓣API获得电影详情信息

void collectMovie();//收藏电影, 调用SubjectModel,访问服务端，完成电影收藏功能，调用SubjectFragment的showCancelCollectButton()方法显示取消收藏按钮。

void cancelCollect();//取消收藏电影，调用SubjectModel,访问服务端，完成取消收藏功能，调用SubjectFragment的showCollectButton()方法显示收藏按钮。

void getCollectOrNot();//判断该电影收藏与否，用于决定显示收藏按钮还是取消收藏按钮。

void addHistory();//增加浏览记录，

void deleteHistory();//删除浏览

SubjectFragment类方法：

void showMessage(String message);//显示信息

void initView(List<String> networkImages,SubjectInfoBean dataBean);//初始化电影详情界面，展示数据

void showCollectButton();//显示收藏按钮

void showCancelCollectButton();//显示取消收藏按钮

void showLoginAct();//展示登录界面

SubjectModel类方法：

public void subjectRequest(String movieID)//获取电影详细信息请求

public void onCollectRequest(String username, String movieID)//收藏电影请求

public void onCancelCollectRequest(String username, String movieID)//取消收藏请求

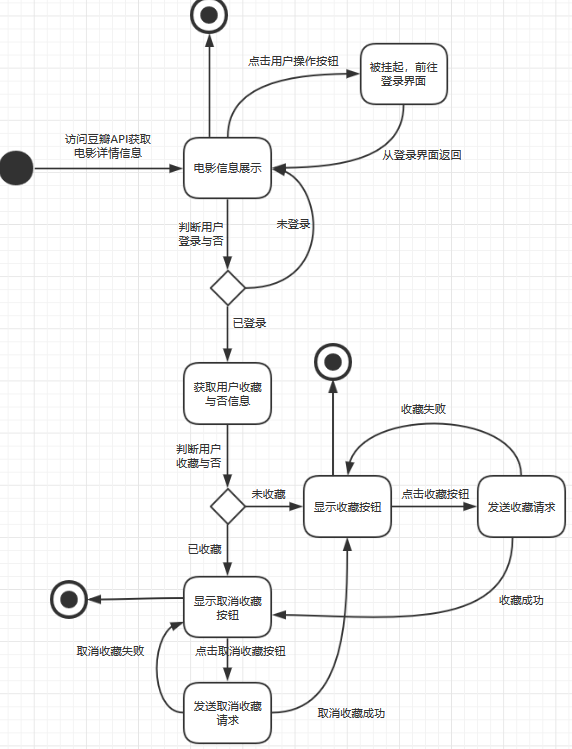
public void onCollectOrNotRequest(String username, String movieID)//判断收藏与否

public void onAddHistoryRequest(String username, String movieID)//增加浏览记录请求

public void onDeleteHistoryRequest(String username, String movieID)//删除浏览记录请求

###### 4.3.8类内，类外方法调用状态图

SubjectView类调用状态图：



###### 4.3.9限制条件及出错处理

网络数据加载应该再10s之内完成，否则会被认定为连接超时放弃连接。如果数据加载失败，可以点击浮动按钮的刷新按钮进行重新加载。

用户进行收藏电影，取消收藏等操作之前需要进行登录，如果未登录，会跳转到登录界面。