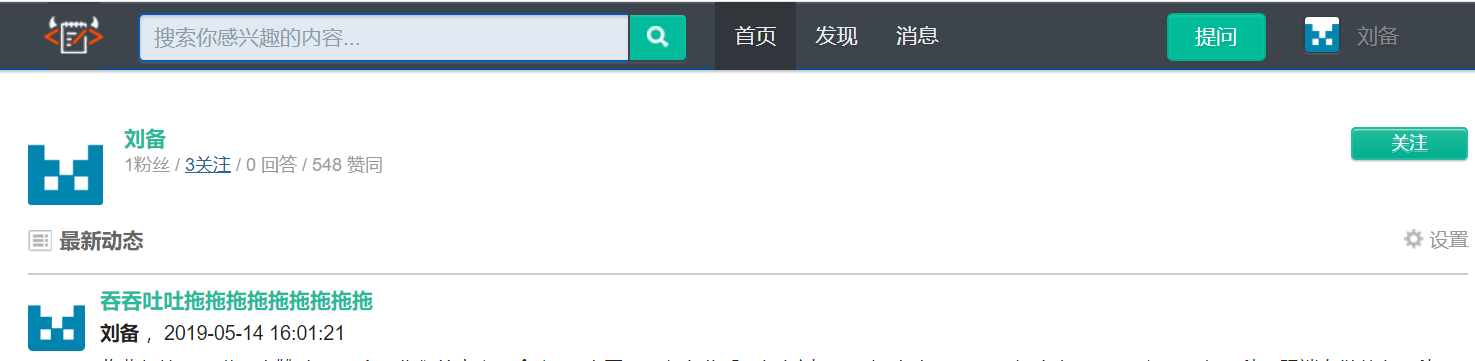
**实现关注功能**

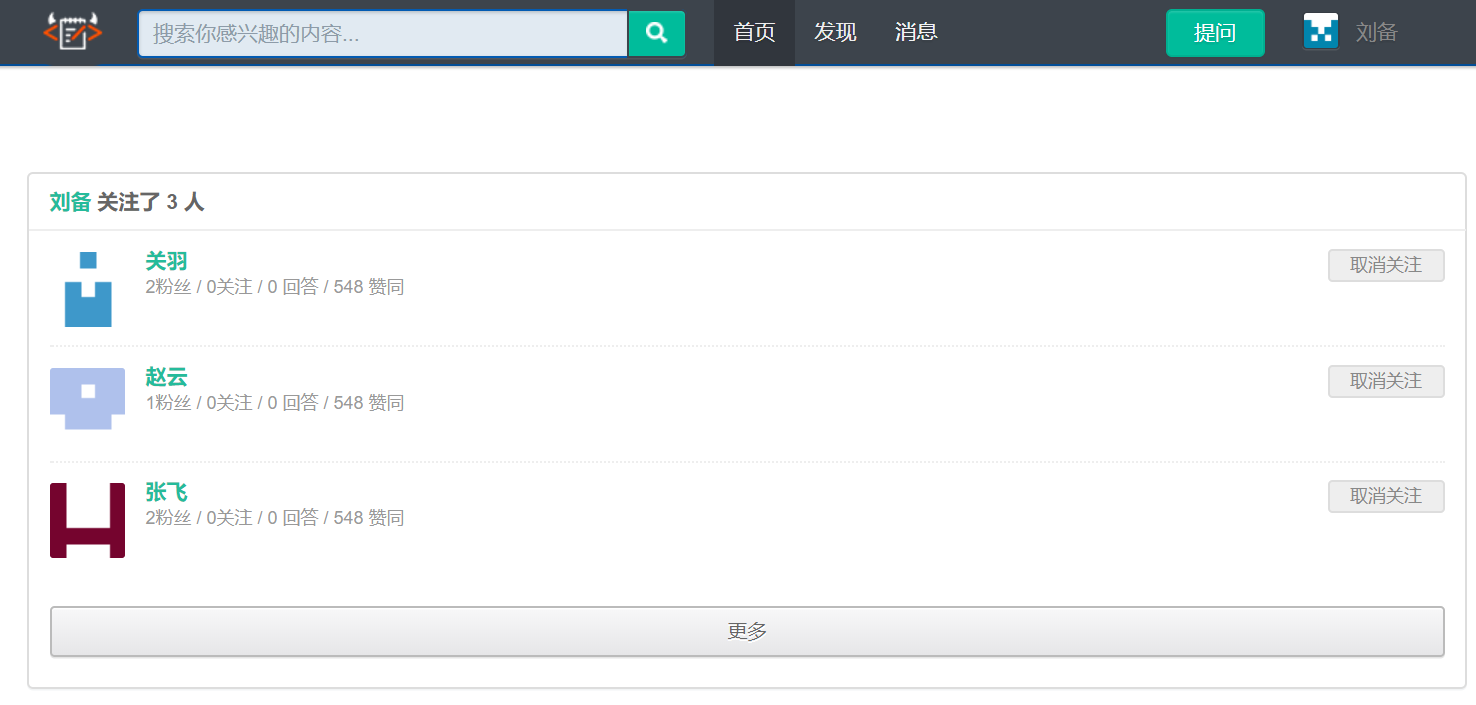
需求分析：

每个用户可以选择关注问题（收藏问题），关注其他用户等操作。有了关注功能后，在每个用户后面就可以增加显示关注数量，粉丝数量等信息（回答数和赞同数还没添加，目前前端的点击关注按钮进行关注还没调好，只能手动输入url进行关注）。同时在问题详情页面可以对问题点击关注，并在下面会显示关注人数和头像，在首页问题前面显示关注数。

个人主页：



自己的关注列表



自己的粉丝列表



问题详情页面



实现概述：

Controller层：

FollowController任务：

1关注用户（取关用户）/followUser和/unfollowUser,以json格式返回

接收前端传来的要关注的userId，首先判断当前是不是登陆用户，不是就返回999状态码提示去登陆，数据库中查这个userId是不是存在，不出在返回状态码1，都没问题就调用service进行关注操作。如果成功返回状态码0和目前自己的已关注人数。（取消关注和这个一样，指示调用service的操作变了）。

2 关注问题（取关问题，关注和收藏是一个意思）/followQuesion和/unfollowQuestion,以json格式返回

接收前端传来的要关注的questionId，首先判断当前是不是登陆用户，不是就返回999状态码提示去登陆，数据库中查这个questionId是不是存在，不出在返回状态码1，都没问题就调用service进行关注操作。然后将关注者的信息（头像，id,用户名）和这个问题的关注数（收藏数）放入map中。最终如果成功返回状态码0和map转化的json。（取消关注和这个一样，指示调用service的操作变了,还有就是map中的信息不要头像和用户名了）。

3 显示关注列表（这里做的是用户列表，用户的问题关注列表还没做，后面可以进行扩展） /user/{uid}/followees

View层需要三个信息

1关注的用户列表，这个根据前端传来的额uid查找自己的关注列表，然后根据表中的id进行用户信息的构造（因为现在这个版本开始用户信息增加了关注数和粉丝数等），这里采用的是函数封装这个转换功能。将根据id查到的user和followerCount等参数进行捆绑。同时followed的这个参数是指示前端要显示“关注”按钮还是“取消关注”按钮（就是告诉前端当前登录用户与列表中的用户的关注关系），显然对于未登录用户则全部都显示成“关注”按钮，因此传入false。

2当前用户头像，根据uid查到自己的user对象传出去

3 我的关注数量，在service中查找。

最后将这三个信息通放入model。



4显示粉丝列表 /user/{uid}/followers这个和显示关注用户基本一致，只是在调用得service不一样。

HomeController任务：在每个问题前面显示对其的关注数量，直接根据questionId调用service层去查找就行。

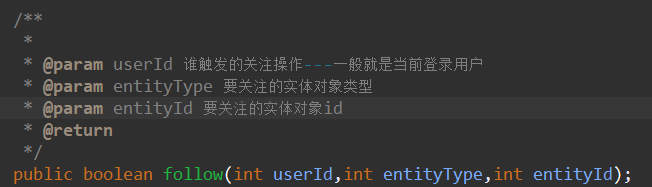
QuestionController任务：问题详情中显示这个问题的关注数量和关注者的头像列表。在原来的基础上调用followServiceImpl处理，获得粉丝id列表，然后取出相应的user，得到头像用户名等信息给view层。



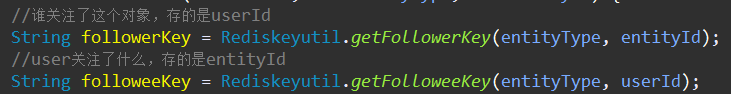
Service层的FollowService功能：

1 关注功能（取消关注）

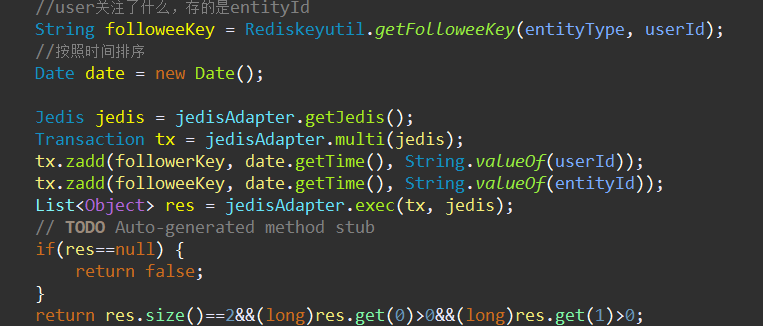
因为一个用户关注的动作需要做两个事情，一个是在这个用户的关注列表中加入要关注的实体id,另一个是要在这个实体的粉丝列表中假如这个用户的id。



首先创建redis的两个key,一个是关注列表的followeeKey，这个key中存就是当前用户关注的对象的id，因此构造的时候以entityType和userId作为key的构造参数，而entityId作为存储的内容，第二个是粉丝列表followerKey,表示这个实体中有多少用户关注了，因此这个存的就是当前用户id，而以entityTpye和entityId组成key。



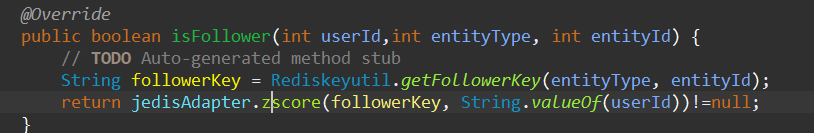
接下来利用redis的事务进行操作，因为这两个任务要一起成功或者一起失败，进行sortedSet的存储的时候用日期来进行排序。事务操作返回的List是每个动作相关，由此来进行最后进行结果验证



2 获取粉丝列表和获取关注列表：构造好对应的key后直接利用zrange进行读取。因为返回的是List<string>，因此写一函数装换成List<Integer>.

3 获取粉丝数数和获取关注数量：构造好对应的key后直接利用zcard进行读取。

4 返回是不是已经关注: 构造好对应的key后直接利用zscore进行读取,这个返回的是传入的userId对应的排序对象（就是时间date的double形式），通过判断是不是为空来确认。



Utils层：这一层里包装好了redis的相关操作，直接给service层调用。

1、JedisAdapter：封装了对redis的直接操作（相当于mapper层的功能），这里目前主要实现了用SortedSet来存放关注列表和粉丝列表。同时采用了redis的事务来操作



2、Rediskeyutil:自定义了redis的key生成工具，方便统一管理。

