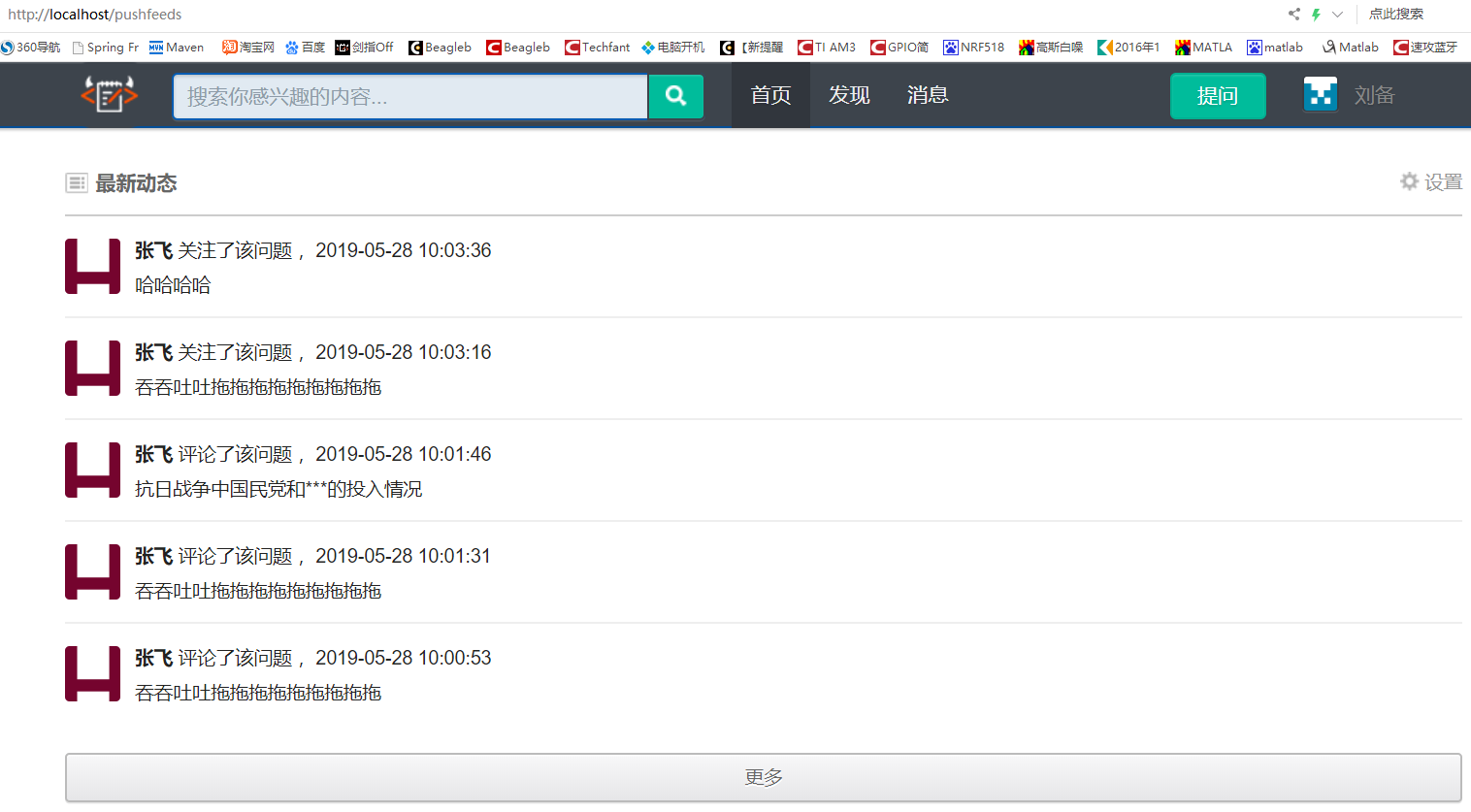
实现timeline功能（用户动态）

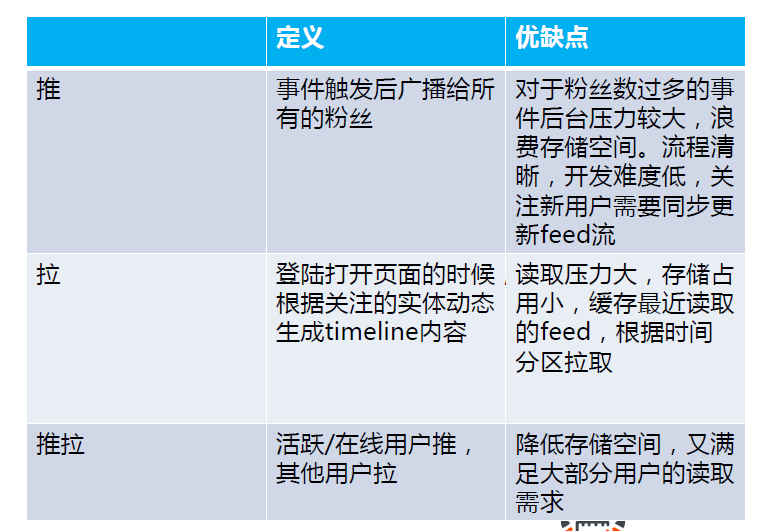
需求分析：

对于每个用户可以查看自己timeline（两种方式的url分别为：pullfeeds、pushfeeds））（未登录用户查看系统timeline—就是所有用户的动作）,里面内容主要包括自己关注用户触发的新鲜事（就是回答、关注、点赞等动作），这里主要实现了关注用户回答问题，关注问题的推送。



实现概述：

目前实现推送主要采用推拉方式，其各自的优缺点，目前分别实现了推和拉两种方式。



推push：当有用户触发了某个新鲜事，首先也是在异步中产生一个feed对象（这个对象中存储了新鲜事相关的核心数据）插入数据库(这些和pull方式可以共用)。然后找到自己的粉丝列表（同时在列表中加入系统用户0），然后遍历列表，利用redis的lpush将自己feed的id插入到每个粉丝自己的timeline队列中（这个过程就是push的过程，这里可以知道系统0的队列中存储了所有的feed-id）。然后粉丝打开自己的timeline页面就直接从redis中取出自己的timeline队列，把里面的feed-id取出，然后取数据库将feed读出来传给前端处理。

拉pull:有用户触发了某个新鲜事，就通过异步方式产生一个feed对象，然后插入数据库中即可。当其他用户打开自己的timeline页面，就会首先从redis中找到自己关注对象的id，然后去数库中找到自己关注对象的所有feed(以list<Feed>形式返回)，将这些对象交给前端处理就行。

Controller层：

FeedController任务：

Push方式：首先获取当前用户id(没有登录给0)，根据这个id从redis中取出自己的timeline队列，获得属于自己的feed-id，然后调用service从数据库中奖feed取出并给前端进行模板渲染。

Pull方式：首先获取当前用户id,然后获取自己的关注对象的userId(放到list中，如果是未登录用户或者没有关注对象list就为空，最后在mapper层会进行判断)，然后调用service将feeds取出给前端渲染。

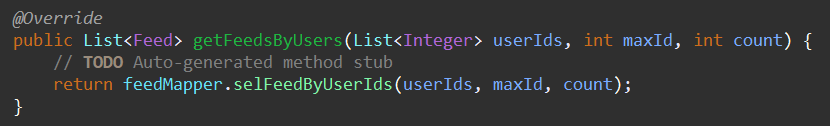
Service层：

FeedService功能：

1新增Feed对象进行过滤，插入到数据库中。

2根据id查到Feed对象

3根据List<Integer> userIds查找自己list中用户的所有feeds.这里还提供了maxId和count两个参数，和分页查找的效果一样，count是要查找的个数，maxId主要是用于分段读取，只读取id小于这个数的feed,因为查feed的时候是按id倒序，因此通过缩小maxid来缩小范围。



# Mapper层：FeedMapper的主要功能

# 1、插入Feed对象，这里要注意的一点是用了Mybatis基于注解的方式获取新增记录的自增长id的功能（因为在handler中插入了feed对象后需要将这个对象自增的id放入粉丝的timeline队列中）。

# 

# 2、根据id返回feed

# 3、根据List<Integer> userIds查找到对应的所有feeds，同事如果userIds是个空的list就查找所有的feeds，因此这里用到了动态sql，采用xml绑定方式。

# 

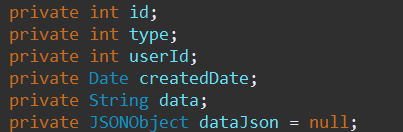
async.handler层：FeedHandler主要负责异步处理的相关操作

1、pull和push方式的共同部分。创建feed对象，将事件的Model的相关信息放入，关键点是data属性的处理，这里要将其组织成json的格式然后再放入。目前实现的是用户回答了问题和用户关注了问题两个新鲜事，因此需要触发者user和question两个对象的相关信息。同时因为关注的还可能是人或者回答，因此这里要指定是关注问题才取出问题，对于其他关注可以在这个函数中进行扩展。

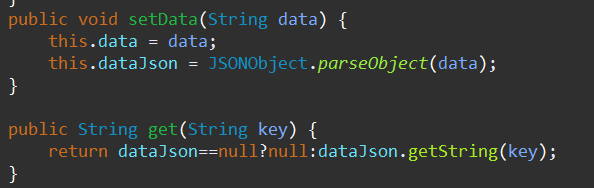


2、对于push方式还应该在这里将自己的feed推送到粉丝队列中，首先获取自己的粉丝id,然后加上0(没有粉丝或者没有登录时候的情况)，根据这个参数获取redis中的队列（如果原来没有，会自动新建），然后将自己的feed-id插入到其队列中（这里虽然在新建feed对象的时候没有setId,但是执行插入操作后会自动获取自增的id,这个在mapper层已经设置过了）。

Model层：Feed对象的设计：其中type用来表示新鲜事类型（事件类型，EventType中的）userId是触发者的id,data中存储了这和时间的核心数据格式为json，因此引入了dataJson进行辅助转换json对象。（数据库中没有）



在插入的时候就转化为json,同时定义了get(Key)函数，这样对象直接传到前端的时候就volecity解析的时候很方便（volecity中调用了 feed.xxx,就会在feed对象中找getxxx()或者get(xxx)这样的函数）



具体实现：

数据库设计：新增feed表

DROP TABLE IF EXISTS `feed`;

CREATE TABLE `feed` (

`id` INT NOT NULL AUTO\_INCREMENT,

`created\_date` DATETIME NULL,

`user\_id` INT NULL,

`data` TINYTEXT NULL,

`type` INT NULL,

PRIMARY KEY (`id`),

INDEX `user\_index` (`user\_id` ASC))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8;

其中包括id，评论内容content,新鲜事发起者user\_id,新鲜事类型 type,创建时间created\_date,事件的核心内容(关键点)。