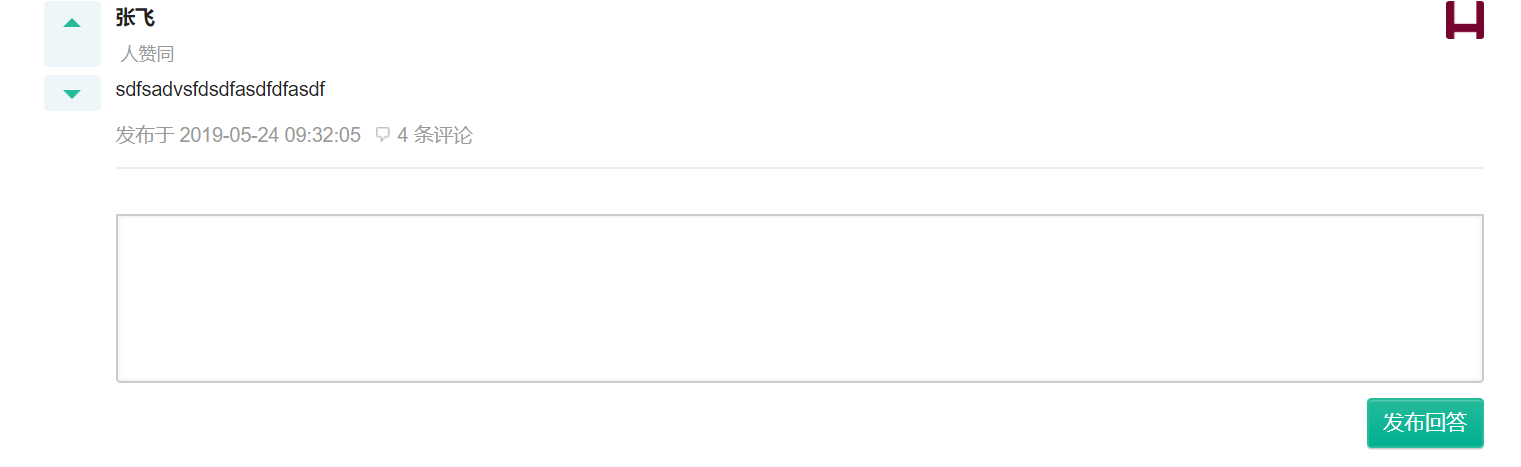
**评论中心和站内信**

**一、评论中心（回答问题）**

需求分析：

1、在问题详情页面中，登陆用户可以对问题进行回答，提交自己的回答给后台url为（“/addComment”），如果不是登陆用户重定向到登陆页面。是登陆用户就进行一些验证（敏感词等），之后提交，然后重定向到问题详情页面



2、在问题详情页面(url为：/question/{qid})，将问题及其以下的所有的回答（回答实体及回答人的信息捆绑）分页显示出来，同时还要更新在首页的回答数量（更新question中的comment\_count就行）。



实现概述：

Controller层：

CommentController任务：1接收前端给的questionId和回答内容两个参数。通过hostHolder状态判断是不是登陆用户，不是就重定向到登陆页面，是的话就创建comment对象，调用Service层进行插入。然后对question中的comment\_count进行更新（之后考虑异步化）。最后重定向回消息详情页面。

QuestionController任务：上次在详情功能上只把问题挂上去，除了问题外，还应该有对这个问题的所有回答。分页找出该qustionid下的的评论List<Comment>，然后根据评论中的userId找到其发布者，进行捆绑，给详情页面进行渲染显示。

Service层：

CommentService功能：1新增的评论对象进行过滤，插入到数据库中。2根据id和type查到对应的问题对象集合返回List<comment>。3、根据id和type查找该问题下的所有评论数（用于更新）--int返回。

说明：因为这里要做成一个评论中心的样子，既可以对问题评论（回答），也可以对回答进行评论，因此只用id来表示评论对象是不够的（如问题的id很可能和某个回答的id是一样的，因此引入entityType来进行二维的区分，在数据库的设计中会有体现）。

Mapper层：CommentMapper:插入Comment对象，分页查找id/type下所有评论，返回id/type下的评论对象数量。

Model层：Comment对象,EntityType(存了一些常用被评论对象的type,方便调用)

具体实现：

数据库设计：新增comment表

DROP TABLE IF EXISTS `comment`;

CREATE TABLE `comment` (

`id` INT NOT NULL AUTO\_INCREMENT,

`content` TEXT NOT NULL,

`user\_id` INT NOT NULL,

`entity\_id` INT NOT NULL,

`entity\_type` INT NOT NULL,

`created\_date` DATETIME NOT NULL,

`status` INT NOT NULL DEFAULT 0,

PRIMARY KEY (`id`),

INDEX `entity\_index` (`entity\_id` ASC, `entity\_type` ASC)

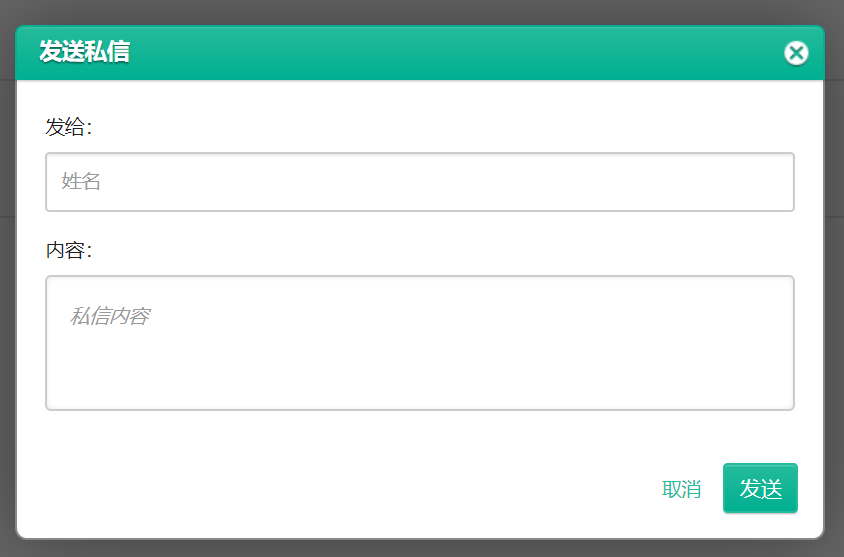
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

其中包括id，评论内容content,发布者user\_id,被评论对象的标识符 entity\_id和entity\_type,创建时间created\_date,状态status(用于删除评论—一般不直接把用户的信息删除掉，而是采用失效的方法来做。

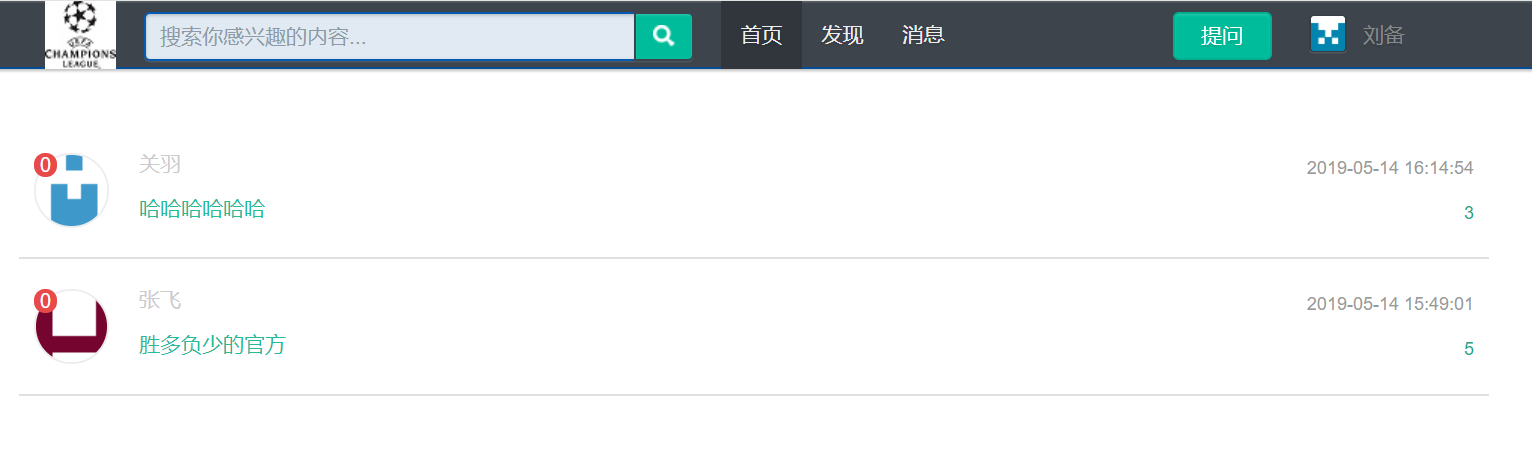
**二、站内信**

需求分析：

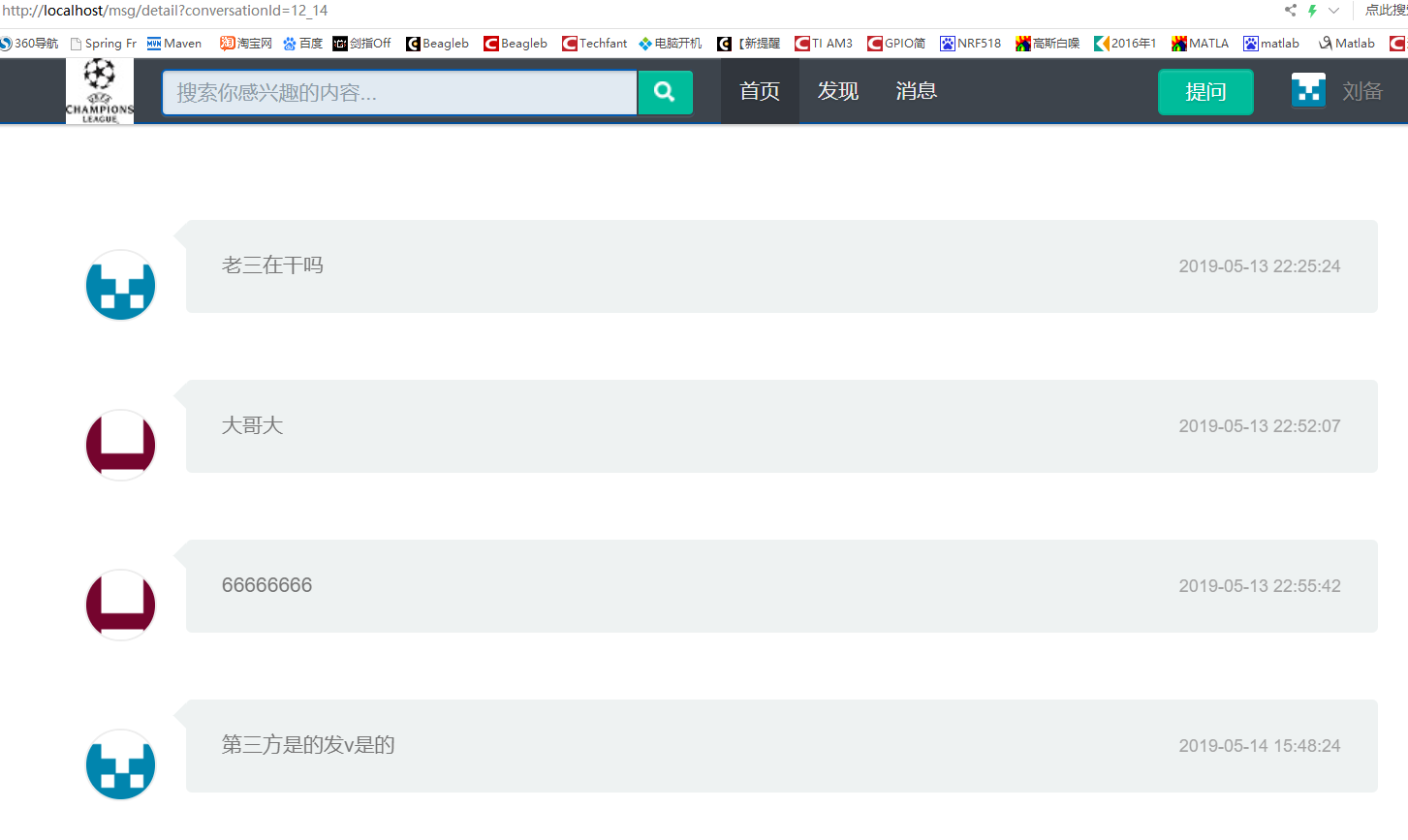
1、通过提交要发送的用户名和内容可以向对方发站内信



2、点击我的私信，显示我的站内信列表（和不同用户的对话，每个用户显示最新的一条），按照时间顺序排下来。并且每个对话显示未读的消息数



3、点击站内信列表中的其中一条可以显示对话详情，同时清空未读信息数



实现概述：

Controller层：MessageController主要有三个功能：

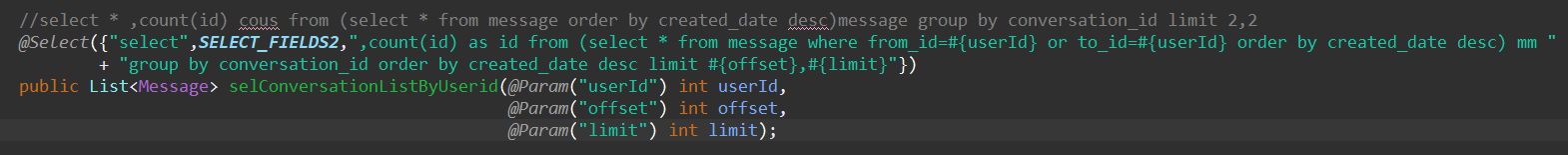
1、发送信息（url: /msg/addMessage）：接收前端传来的对方用户名和消息内容，首先判断是不是登陆用户，不是的话返回状态码999给前端，前端会自动跳转到登陆页面，如果是登陆用户就通过用户名查询要发送的对方用户是否存在，不存在返回状态码1和错误信息，如果都没问题就创建Message对象并交给service插入，这里约点conversationid为发送合接收的两者id较小的排前面。最终成功范湖状态码0.（返回的都是json串形式）

2、个人会话列表显示（url: /msg/list）：用户访问“msg/list”,如果不是登陆用户重定向到登陆页面，是登陆用户就从数据库中查找当前userId参与的所有会话，与每个用户的会话只显示会话的最新一条消息和未读数量，通过查到的messaged对象可以获得对方的userId，然后获取对方的User对象用于显示头像用户名等。最后在每个会话中查找toid为当前id,hasRead为0的数量，将三者捆绑放入model进行渲染返回。

3、会话详情显示(url:/ msg/detail)：接收来来自前端的conversationId查找会话的所有消息信息List<Message>,根据Message中的formId找到发起者，跟每条消息捆绑放入model.渲染出详情页面返回给前端。同时进入个会话后将toId为当前用户的（对方发过来）的未读标志位全部置位1，清空未读数。

Service层：MessageService主要功能：1对发的站内信内容进行过滤，插入到数据库中。2获取会话详情。3 获取会话列表。4 获取会话未读数。5 清空会话未读数

Mapper层：MessageMapper：其他的功能和Service对应，主要是分页查找会话列表比较复杂。



首先对表进行按照时间排序（最近的放前面），然后根据conversationId进行分组，查找标的所有字段，因为id是没有用的，而且还需要每个会话的总的对话数，因此将用id作为别名来存储这个数据，最后整个过程按时间排序（最近的会话放在前面）。

Model层：Message对象

具体实现：

数据库设计：新增message表

DROP TABLE IF EXISTS `message`;

CREATE TABLE `message` (

`id` INT NOT NULL AUTO\_INCREMENT,

`from\_id` INT NULL,

`to\_id` INT NULL,

`content` TEXT NULL,

`created\_date` DATETIME NULL,

`has\_read` INT NULL,

`conversation\_id` VARCHAR(45) NOT NULL,

PRIMARY KEY (`id`),

INDEX `conversation\_index` (`conversation\_id` ASC),

INDEX `created\_date` (`created\_date` ASC))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8;

其中包括id,发起人id:from\_id,接收人id:to\_id,消息内容 content,创建时间created\_date,读取状态has\_read,会话id:conversation\_id。