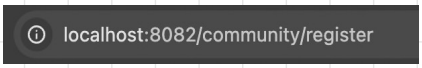
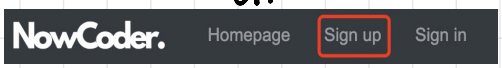


or.

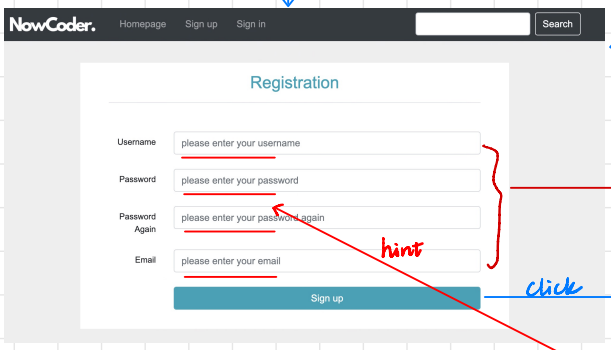


DispatcherServlet

Login Controller

```
@RequestMapping(path = "/register", method = RequestMethod.GET)
public String getRegisterPage() {
    return "/site/register";
}
```

= resources/templates/site/register.html (Thymeleaf)



Automatic mapping: <input name="*"> match == entity User.set*( )

Spring MVC Auto-Binding maps HTTP data to Java objects via setters.

hint

click

```
@RequestMapping(path = "/register", method = RequestMethod.POST)
public String register(Model model, User user) {
```

UserService

```
public Map<String, Object>
register(@ModelAttribute("user") User user) {
```

① if user == NULL → throw IllegalArgumentException

② if input field empty?

③ if existing username/password? → map.put("xxxMsg","xxx");

④ register →

```
user.setSalt(CommunityUtil.generateUUID().substring(0, 5));
user.setPassword(CommunityUtil.md5(key: user.getPassword() + user.getSalt()));
user.setType(0); // 1 = normal user, 2 = admin, 3 = author/poster
user.setStatus(0); // 0 = unactivated, 1 = activated
user.setActivationCode(CommunityUtil.generateUUID());//generate activationcode
user.setHeaderUrl(String.format("http://images.nowcoder.com/head/%dt.png", new
user.setCreateTime(new Date());
userMapper.insertUser(user);
```

⑤ send activation email

```
mailClient.sendMail(user.getEmail(), "Please activate your
account - NowCoder", content);
        String content = templateEngine.process("/
        mail/activation", context);
```

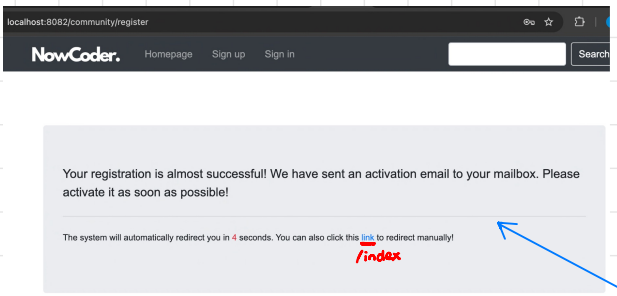
html Template ← user.email + activation url

return map with error message

model.addAttribute()

return null

```
return "/site/operate-result";
```



topic

Please activate your account - NowCoder

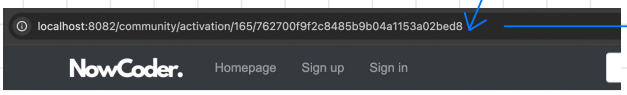
nowcoderofficial@sina.com
Today, 16:47
Chutong Ren
Reply all

user.email

chutong.ren@tum.de, Hello!

You are registering on NowCoder. This is an activation email. Please click click here, to activate your NowCoder account!

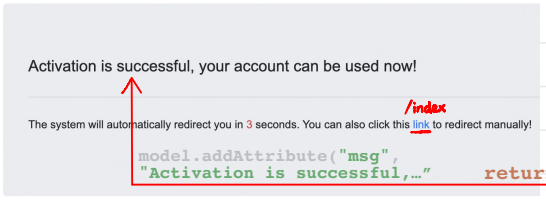
activation url




```
@RequestMapping(path = "/activation/{userId}/{code}", method =
RequestMethod.GET)
public String activation(Model model, @PathVariable("userId")
int userId, @PathVariable("code") String code) {
```

UserService

```
public int activation(int userId, String code) {
    User user = userMapper.selectById(userId);
    if (user.getStatus() == 1){
        return ACTIVATION_REPEAT;
    }else if(user.getActivationCode().equals(code))
{
        userMapper.updateStatus(userId, 1);
        clearCache(userId);
        return ACTIVATION_SUCCESS;
    }else{
        return ACTIVATION_FAILURE;
    }
}
```



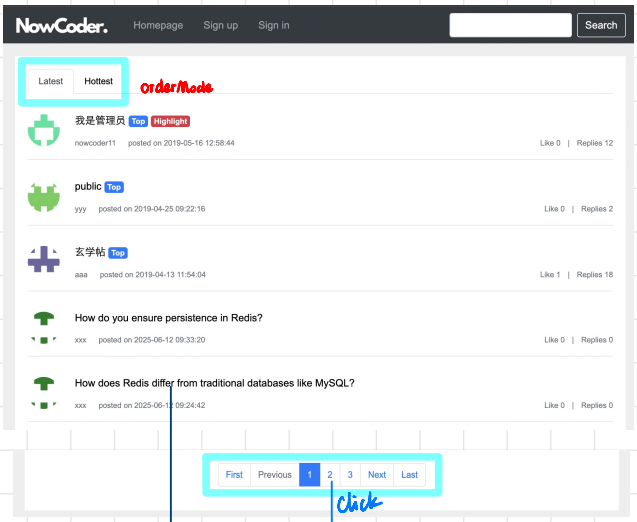

```
model.addAttribute("msg",
"Activation is successful,…")
```

```
return "/site/operate-result";
```

id	username	password	salt	email	type	status	activation_code
162	xiuzhi	119178ca86df3e3b01ed7c0cce16…	12d95	15145661733@163.com	0	0	fb9862658eab4970aaa0e555a508c6a0
165	chutong	b54cb6372a8f76870428a5976b55…	a52cd	chutong.ren@tum.de	0	1	762700f9f2c8485b9b04a1153a02bed8
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

HomeController

localhost:8082/community/index



Click

localhost:8082/community/index?orderMode=1&current=2

page.setCurrent(2)

offset = (current-1) * limit

```
@RequestMapping(path = "/index", method = RequestMethod.GET)
public String getIndexPage(Model model, Page page,
@RequestParam(name = "orderMode", defaultValue = "0") int
orderMode) {

    List<DiscussPost> list = discussPostService
            .findDiscussPosts(0, page.getOffset(), page.getLimit(),
    orderMode);

    List<Map<String, Object>> discussPosts

    model.addAttribute("discussPosts", discussPosts);
    model.addAttribute("orderMode", orderMode);
    return "/index";
}
```

userId, postId → user, likeCount + post

→ Style active?

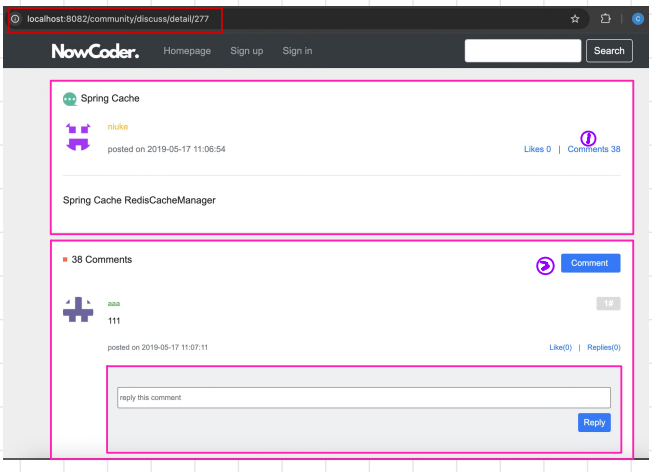
```
<a th:class="|nav-link ${orderMode==0?'active':''}|"
th:href="@{/index(orderMode=0)}"> Latest</a>
```

```
public class Page
page.setRows(discussPostService.findDiscussPostRows(0));
page.setPath("/index?orderMode=" + orderMode);
```

点击 → 变更url(后拼接3-个id) → 再发执行该url映射的函数.

```
<a th:href="@{|/discuss/detail/${map.post.id}|}"
th:utext="${map.post.title}">111</a>
```

DiscusspostController




```
@RequestMapping("/discuss")
public class DiscussPostController implements CommunityConstant {

@RequestMapping(path = "/detail/{discussPostId}", method =
RequestMethod.GET)
public String getDiscussPost(@PathVariable("discussPostId") int
discussPostId, Model model, Page page) {

    DiscussPost post =
    discussPostService.findDiscussPostById(discussPostId);
    model.addAttribute("post", post);

    User user = userService.findUserById(post.getUserId());
    model.addAttribute("user", user);

    List<Comment> commentList =
    commentService.findCommentsByEntity(
            ENTITY_TYPE_POST, post.getId(),
    page.getOffset(), page.getLimit());
```

CommentVoList → "comment": { "user": , "likeCount": , "likeStatus": , "replys": , "replyCount": }

replys → replyVoList → "reply": , "user": , "target": , "likeCount": , "likeStatus":

```
List<Comment> replyList =
commentService.findComment
(ENTITY_TYPE_COMMENT,com
),0, Integer.MAX_VALUE);

    model.addAttribute("comments", commentVoList);

    return "/site/discuss-detail";
}
```

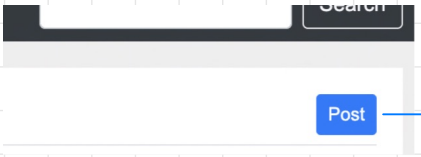
Comment

id	user_id	entity_type	entity_id	target_id	content	status	create_time
232	159	1 post	281	0	NIHAO	0	2025-04-01 22:25:45
233	159	2 comment	232	0	HAHA	0	2025-04-01 22:25:52

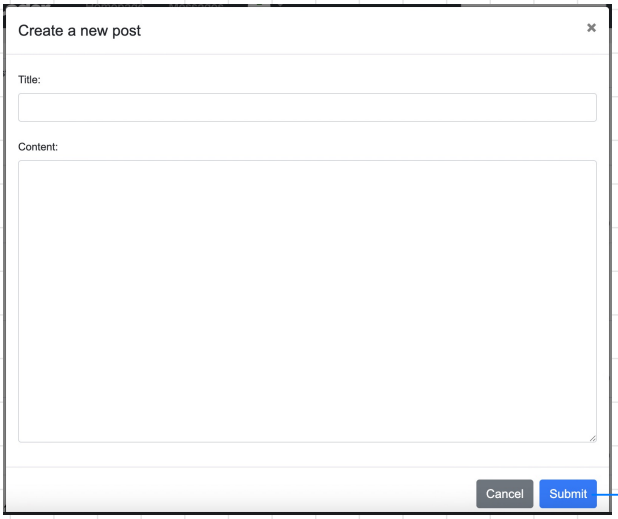
in a comment, reply to whom? (userId)

offset → the row number (start)

limit

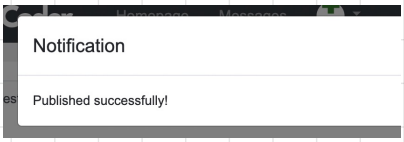



```
@RequestMapping(path = "/add", method = RequestMethod.POST)
@ResponseBody
public String addDiscussPost(String title, String content) {
```

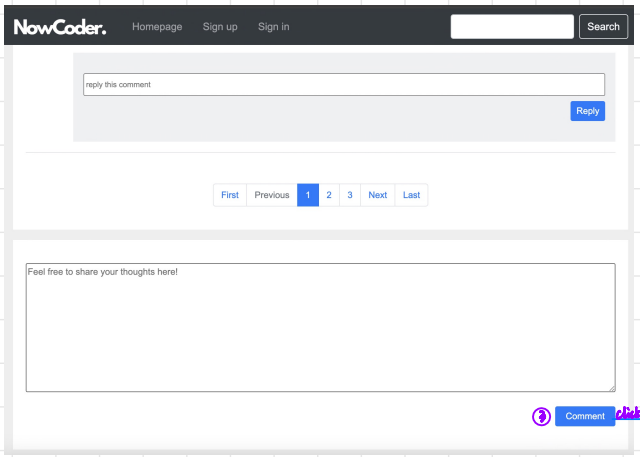



```
$(function(){
    $("#publishBtn").click(publish);
});

function publish() {
    $("#publishModal").modal("hide");

    var title = $("#recipient-name").val();
    var content = $("#message-text").val();

    $.post(
        CONTEXT_PATH + "/discuss/add",
        {"title":title,"content":content},
        function(data) {
            data = $.parseJSON(data);

            $("#hintBody").text(data.msg);
            $("#hintModal").modal("show");

            setTimeout(function(){
                $("#hintModal").modal("hide");

                if(data.code == 0) {
                    window.location.reload();
                }
            }, 2000);
        }
    );
}
```

```
@Transactional(isolation =
Isolation.READ_COMMITTED, propagation =
Propagation.REQUIRED)
public int addComment(Comment comment) {
```



then you can see this post in homepage without reloading/refreshing the page




```
@RequestMapping(path = "/add/{discussPostId}", method =
RequestMethod.POST)
public String addComment(@PathVariable("discussPostId")
int discussPostId, Comment comment) {
    commentService.addComment(comment);
```

Declarative Transaction Management → either all succeed or all roll back

```
@Transactional(isolation = Isolation.READ_COMMITTED,
propagation = Propagation.REQUIRED)
public int addComment(Comment comment) {
    comment.setContent(HtmlUtils.htmlEscape(comment.getContent(
    )));
    comment.setContent(sensitiveFilter.filter(comment.getConten
    t()));
    int rows = commentMapper.insertComment(comment);

    if (comment.getEntityType() == ENTITY_TYPE_POST) {
        int count =
    commentMapper.selectCountByEntity(comment.getEntityType(),
    comment.getEntityId());

    discussPostService.updateCommentCount(comment.getEntityId()
    , count);
        }

        return rows;
    }
```

```
<form method="post" th:action="@{/comment/
add/${post.id}}">
```

id	user_id	title	content	type	status	create_time	comment_count	score
277	149	Spring Cache	Spring Cache Redi...	0	0	2019-05-17 11:06:54	38	1752.57...

Likes 1 | Comments 13

```html
<a href="javascript:;" th:onclick="|like(this,1,${post.id},${post.userId},${post.id});|" class="text-primary">
    <b th:text="${likeStatus==1?'Liked':'Likes'}">Likes</b> <i th:text="${likeCount}">0</i>
</a>
```

```javascript
function like(btn, entityType, entityId, entityUserId, postId) {
    $.post(
        CONTEXT_PATH + "/like",
        {"entityType":entityType,"entityId":entityId, "entityUserId":entityUserId, "postId":postId},
        function(data) {
            data = $.parseJSON(data);
            if(data.code == 0) {
                $(btn).children("i").text(data.likeCount);
                $(btn).children("b").text(data.likeStatus==1?'Liked':"Like");
            } else {
                alert(data.msg);
            }
        }
    );
}
```

```java
@Controller
public class LikeController implements CommunityConstant {
    @Autowired
    private RedisTemplate redisTemplate;

    @RequestMapping(path = "/like", method = RequestMethod.POST)
    @ResponseBody
    public String like(int entityType, int entityId, int entityUserId, int
postId) {
         likeService.like(user.getId(), entityType, entityId, entityUserId);

         long likeCount = likeService.findEntityLikeCount(entityType, entityId);
         int likeStatus = likeService.findEntityLikeStatus(user.getId(), entityType,
         entityId);

         Map<String, Object> map = new HashMap<>();
         map.put("likeCount", likeCount);
         map.put("likeStatus", likeStatus);

          return CommunityUtil.getJSONString(0, null, map);
         }
```

```java
public class LikeService {

    @Autowired
    private RedisTemplate redisTemplate;

    public void like(int userId, int entityType, int entityId, int entityUserId) {

        redisTemplate.execute(new SessionCallback(){
            @Override
            public Object execute(RedisOperations operations) throws DataAccessException {
                String entityLikeKey = RedisKeyUtil.getEntityLikeKey(entityType, entityId);
                String userLikeKey = RedisKeyUtil.getUserLikeKey(entityUserId);
                boolean isMember = redisTemplate.opsForSet().isMember(entityLikeKey, userId);

                operations.multi();

                if (isMember) {
                    redisTemplate.opsForSet().remove(entityLikeKey, userId);
                    redisTemplate.opsForValue().decrement(userLikeKey);
                }else{
                    redisTemplate.opsForSet().add(entityLikeKey, userId);
                    redisTemplate.opsForValue().increment(userLikeKey);
                }
                return operations.exec();
            }
        });

    }
```

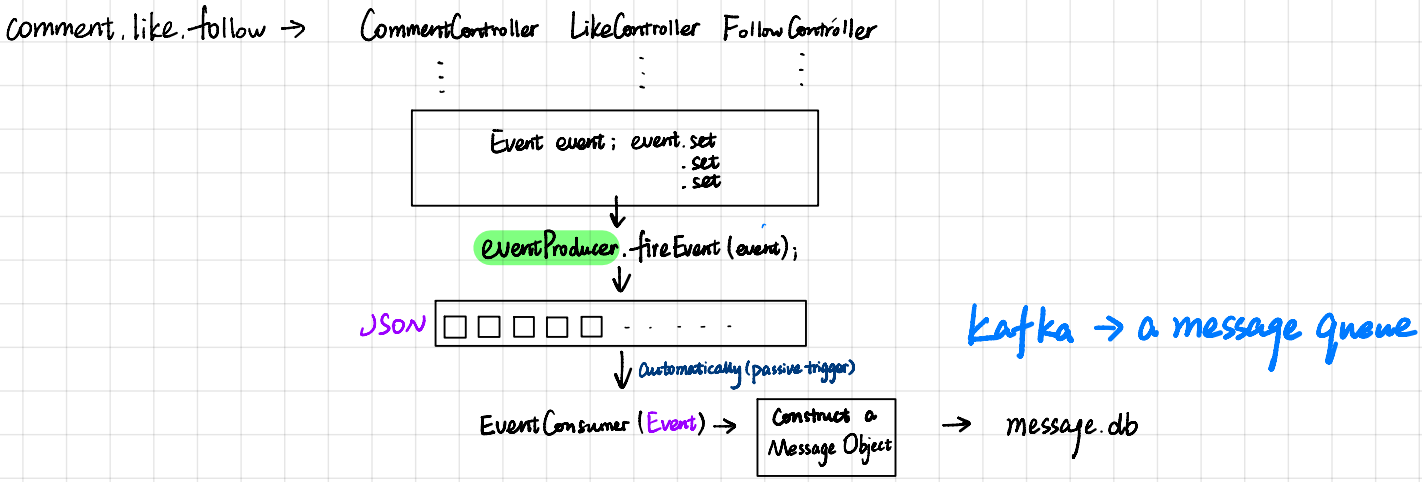
like:entity:1:123

[101,102,103]

判 this user exists in the value → liked

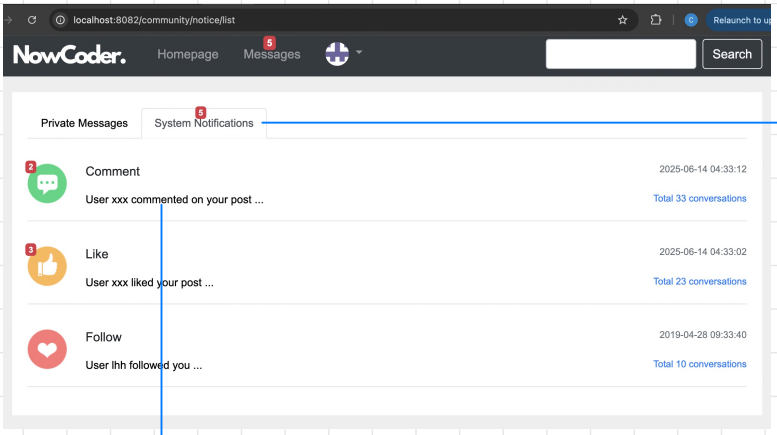
集合中是否存在该值

Redis Transaction Management



id	from_id	to_id	conversation_id	content	status	create_time
202	1	145	like	{"entityType":1,"entityId":273,"postId":273,"userId":111}	1	2019-04-28 12:56:03
203	1	145	comment	{"entityType":1,"entityId":273,"postId":273,"userId":111}	1	2019-04-28 12:56:14
204	1	145	follow	{"entityType":3,"entityId":145,"userId":111}	1	2019-04-28 12:56:18
205	1	111	comment	{"entityType":1,"entityId":272,"postId":272,"userId":145}	1	2019-05-06 05:37:28
206	145	111	111_145	hello	1	2019-05-06 05:37:39

```
String content = HtmlUtils.htmlUnescape(message.getContent());
Map<String, Object> data = JSONObject.parseObject(content, HashMap.class);

messageVO.put("user", userService.findUserById((Integer) data.get("userId")));
messageVO.put("entityType", data.get("entityType"));
messageVO.put("entityId", data.get("entityId"));
messageVO.put("postId", data.get("postId"));
```




```
<a th:href="@{/notice/list}">
```

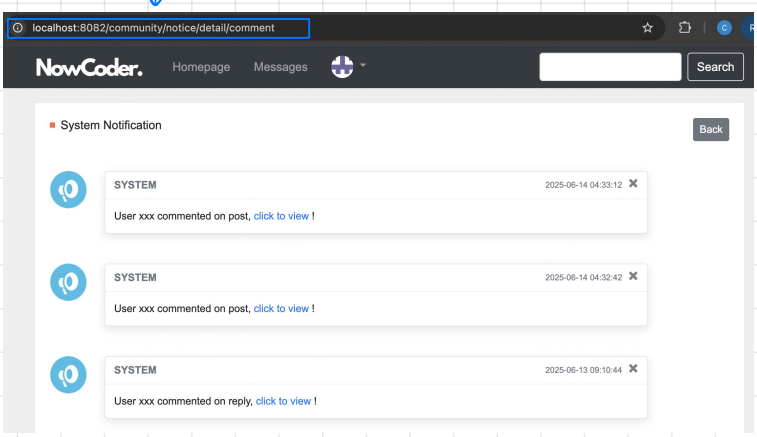
Message Controller

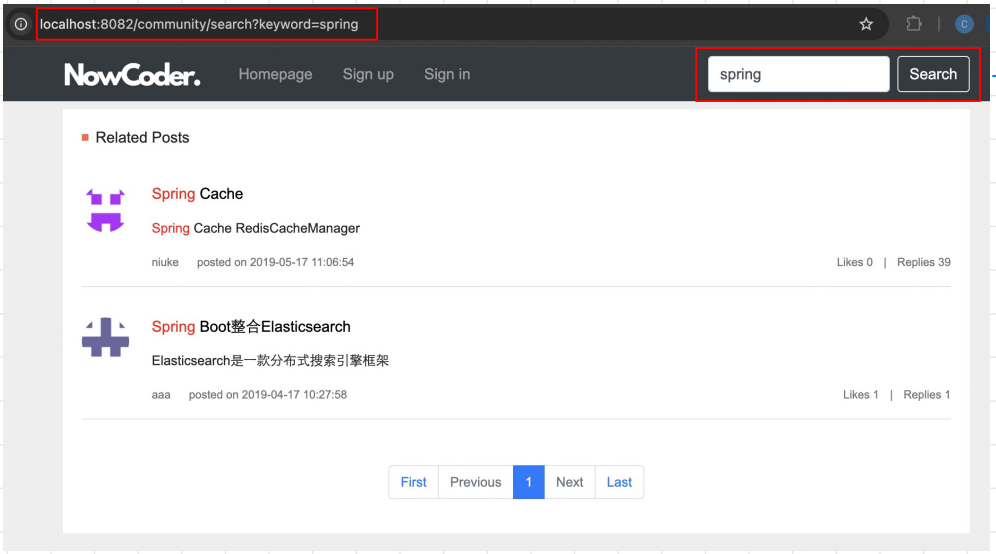
```
@RequestMapping(path = "/notice/list", method = RequestMethod.GET)
public String getNoticeList(Model model) {
    User user = hostHolder.getUser();

    Message message = messageService.findLatestNotice(user.getId(
    int count = messageService.findNoticeCount(user.getId(), TOPI
    messageVO.put("count", count);

    int unread = messageService.findNoticeUnreadCount(user.getId(
    messageVO.put("unread", unread);
```

messageVO

```
{
  "message": {
    "id": 202,
    "fromId": 1,
    "toId": 145,
    "conversationId": "comment",
    "content": "{\"entityType\":1,\"entityId\":273,\"postId\":273,\"userId\":111}",
    "status": 1,
    "createTime": "2025-06-11T10:00:00"
  },
  "user": {
    "id": 111,
    "username": "nowcoder111",
    "headerUrl": "/images/profile111.png"
  },
  "entityType": 1,
  "entityId": 273,
  "postId": 273,
  "count": 1,
  "unread": 1
}
```

```
    model.addAttribute("commentNotice", messageVO);
```





Elasticsearch

① data: MySQL —store to→ es

→ entity/DiscussPost

table and fields mapping

```
@Document(indexName = "discusspost")
public class DiscussPost{

@Field(type = FieldType.Text,
analyzer = "ik_max_word",
searchAnalyzer = "ik_smart")
private String title;
```

→ Services/ElasticsearchService

CRUD functions

```
public void saveDiscussPost(DiscussPost post) {
public void deleteDiscussPost(int id) {
public Map<String, Object>
searchDiscussPost(String keyword, int current,
int limit) throws IOException {
```

↳ dao/elasticsearch/DiscussPostRepository

↓

extends ElasticsearchRepository

→ when add a post,

MySQL → Kafka → es

DiscussPostController (EventProducer)

```
Event event = new Event()
        .setTopic(TOPIC_PUBLISH)
        .setUserId(user.getId())
        .setEntityType(ENTITY_TYPE_POST)
        .setEntityId(post.getId());
eventProducer.fireEvent(event);
```

EventConsumer (@KafkaListener) → elasticsearchService.saveDiscussPost(post);

② Search the "keyword" in es

SearchController

↳ ElasticsearchService → Map<String, Object> searchDiscussPost(keyword)

↳ List<DiscussPost>

↓

search.html

user.type

AUTHORITY_USER = "user" 0

AUTHORITY_ADMIN = "admin" 1

AUTHORITY_MODERATOR = "moderator" 2

```
"/discuss/delete",
"/data/**",
"/actuator/**"
```

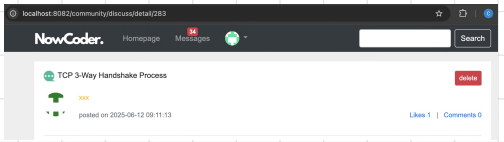
```
"/discuss/top",
"/discuss/wonderful"
```

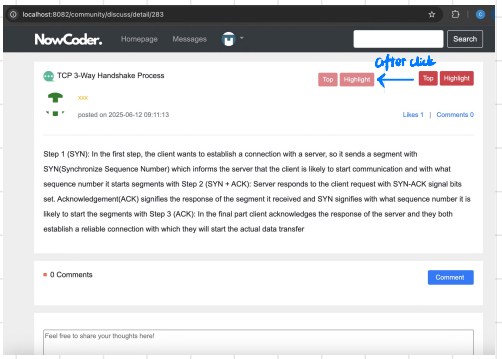
```
"/user/setting",
"/user/upload",
"/discuss/add",
"/comment/add/**",
"/letter/**",
"/notice/**",
"/like",
"/follow",
"/unfollow"
```

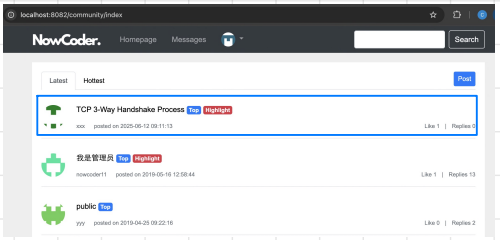
only when logged in

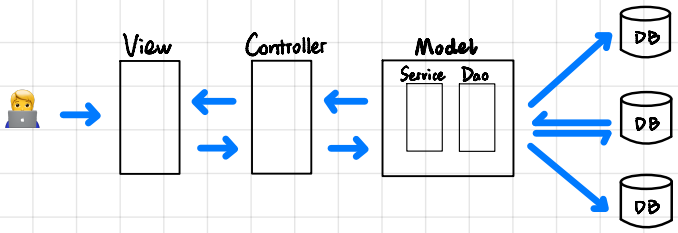
Spring Security

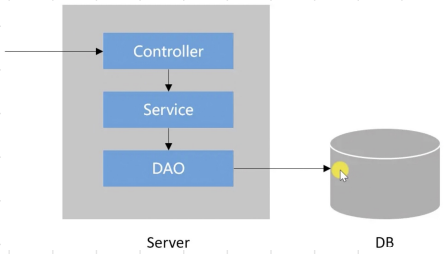
id	usern...	password	salt	email	type	status
21	nowcoder21	25ac0a2e8...	49f10	nowcoder21@sina.com	2 Top.Highlight	1
13	nowcoder13	25ac0a2e8...	49f10	nowcoder13@sina.com	1	1
12	nowcoder12	25ac0a2e8...	49f10	nowcoder12@sina.com	1 delete	1
11	nowcoder11	25ac0a2e8...	49f10	nowcoder11@sina.com	1	1
144	zzz	07b83b774...	21e8b	nowcoder144@sina.com	0	1











5。Controller 不能直接被执行，它是通过 HandlerAdapter 来间接调用的（就像一个适配器）。

⚠️此时会先执行所有拦截器的 preHandle() 方法（相当于“进入 Controller 前”）。