http://8.217.11.251

Tools Version:
JDK 23.0.2 + apache-maven-3.9.9 + Spring Boot 3.4.3 + Spring Security 6.4.3 + MyBatis 3.0.4 + Redis 7.2.7 + Kafka 3.9.0 + Elasticsearch 7.17.1 + MySQL 9.2.0 + kaptcha 2.3.2 + Spring Email 3.4.3 + Quartz + caffeine + Spring Boot Actuator

Run this project in your localhost:

wenn you install all the tools, connect to your database, then run these four commands in four sperate terminals:

brew services start redis

zookeeper-server-start /opt/homebrew/etc/zookeeper/zoo.cfg

kafka-server-start /opt/homebrew/etc/kafka/server.properties

(base) chutongren@ChutongdeMacBook-Pro elasticsearch-7.17.1 % ./bin/elasticsearch

Funtionalities:

Login: Email-based registration with activation email, Secure password storage using MD5 with salted hashing, Login with randomized CAPTCHA verification, Session management to maintain login state, Dynamic UI rendering (different views for guests and logged-in users)

About me: My Profile (displays number of Following (0), Followers (1), and Likes (3)), My posts, Settings (upload new avatar, change Password), Log out

Post: Create posts with sensitive word filtering, Private messaging system, Post list sorted by latest or hottest, View post details, Pagination for browsing posts

Social Features: Like mechanism, Follow/unfollow users, Real-time system notifications (when someone likes, comments, replies, or follows you)

Messages: View and send comments/messages, Comment/message list, Display number of unread items (status-based)

MySQL

Even if it is added to the environment variable, the command initialization must be typed into bin.

mysqld --initialize --console

```
ChutongdeMacBook-Pro:bin root# sudo ./mysqld --initialize --console
2025-02-23T21:56:27.897615Z 0 [System] [MY-015017] [Server] MySQL Server Initial
ization - start.
2025-02-23T21:56:27.898730Z 0 [Warning] [MY-013711] [Server] Manifest file '/usr
/local/mysql-8.4.4-macos15-arm64/bin/mysqld.my' is not read-only. For better sec
urity, please make sure that the file is read-only.
2025-02-23T21:56:27.899241Z 0 [System] [MY-013169] [Server] /usr/local/mysql-8.4
.4-macos15-arm64/bin/mysqld (mysqld 8.4.4) initializing of server in progress as
 process 97051
2025-02-23T21:56:27.901227Z 0 [Warning] [MY-010159] [Server] Setting lower_case_
table_names=2 because file system for /usr/local/mysql-8.4.4-macos15-arm64/data/
 is case insensitive
2025-02-23T21:56:27.910568Z 1 [System] [MY-013576] [InnoDB] InnoDB initializatio
n has started.
2025-02-23T21:56:28.035152Z 1 [System] [MY-013577] [InnoDB] InnoDB initializatio
n has ended.
2025-02-23T21:56:29.061629Z 6 [Note] [MY-010454] [Server] A temporary password i
s generated for root@localhost: s-kaXigp(3=9
2025-02-23T21:56:29.401854Z 0 [System] [MY-013172] [Server] Received SHUTDOWN fr
om user <via user signal>. Shutting down mysqld (Version: 8.4.4).
2025-02-23T21:56:30.165174Z 0 [System] [MY-015018] [Server] MySQL Server Initial
ization - end.
```

A temporary password is generated for root@localhost: s-kaXigp(3=9

Login + reset password

```
C:\Users\Administrator>mysql -uroot -p
Enter password: ************
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.16

Copyright (c) 2000, 2019, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> alter user root@localhost identified by 'lihonghe';
Query OK, 0 rows affected (0.01 sec)
```

brew install mysql

mysql_secure_installation 下载成功，配置密码

Error resolution: https://juejin.cn/post/6844903831298375693

mysql.server start # Start MySQL first

mysql -u root -p # Log in to MySQL 2025@MySQL

You install MySQL on macOS with Homebrew, and you don't need the my.ini file. On macOS and Linux, the MySQL configuration file is my.cnf, not my.ini on Windows.

create database

```
mysql> create database community;
Query OK, 1 row affected (0.00 sec)
```

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| community          |
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.02 sec)
```

create schema(tables) in database

```
mysql> use community;
Database changed
mysql> source /Users/chutongren/Documents/job/nowcoder/community-init-sql-1.5/init_schema.sql
Query OK, 0 rows affected (0.01 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)
```

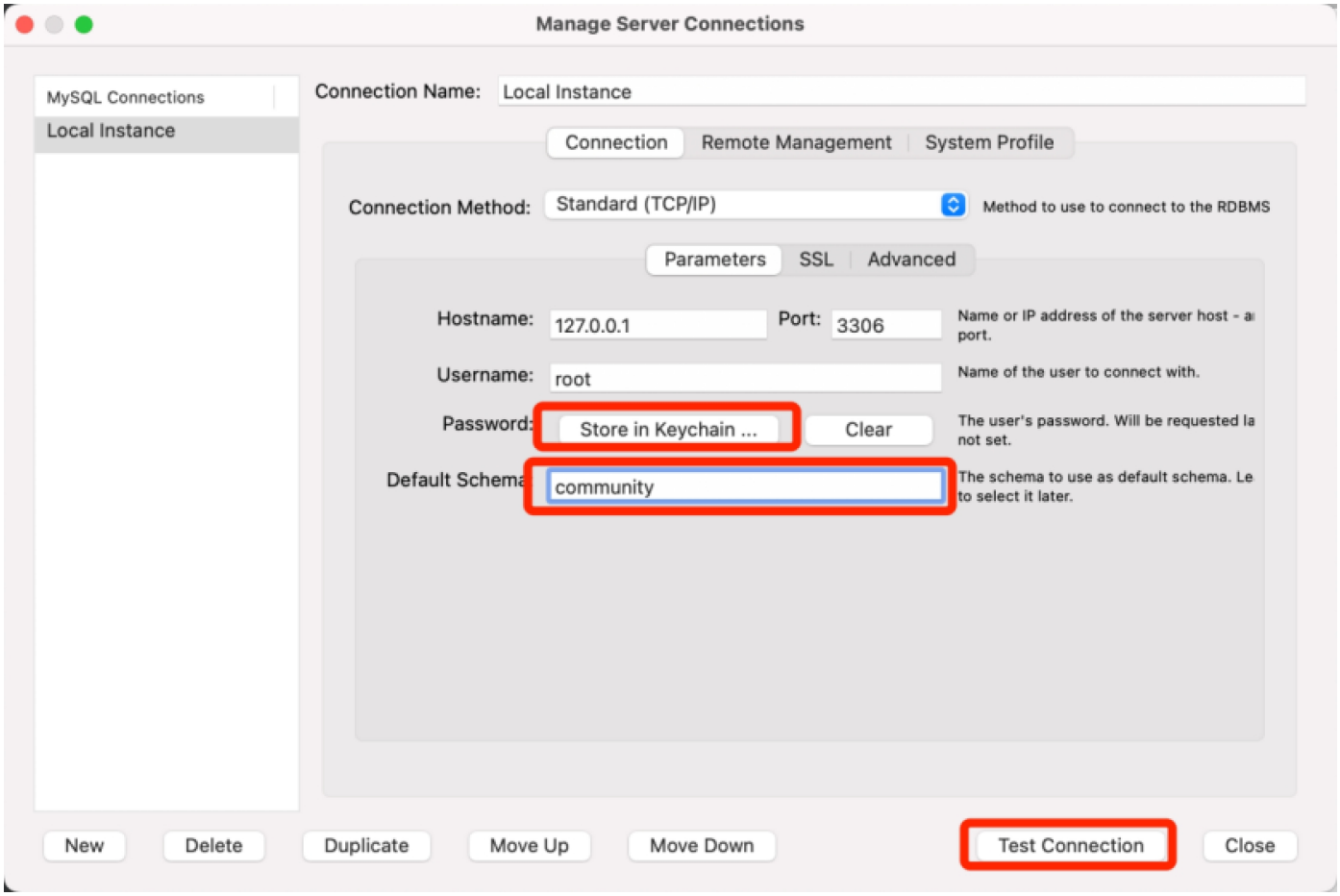
```
mysql> show tables;
+--------------------+
| Tables_in_community |
+--------------------+
| comment            |
| discuss_post       |
| login_ticket       |
| message            |
| user               |
+--------------------+
5 rows in set (0.00 sec)
```

import init_data.sql

```
mysql> source /Users/chutongren/Documents/job/nowcoder/community-init-sql-1.5/init_data.sql;
Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)
```

```
mysql> select * from user;
+-----+-----------+----------------------------------+--------+---------------
------+------+--------+----------------------------------+--------------------
----------------------------+---------------------+
| id  | username  | password                         | salt   | email
      | type | status | activation_code                  | header_url
                           | create_time         |
+-----+-----------+----------------------------------+--------+---------------
------+------+--------+----------------------------------+--------------------
----------------------------+---------------------+
```



Login Modul

Send Activation Email automatically

Spring Email(https://mvnrepository.com/search?q=Spring+Mail)

1.Enable POP3/SMTP service in the mailbox

2.Config：# MailProperties

```
# MailProperties
spring.mail.host = smtp.sina.com
spring.mail.port= 465
spring.mail.username=nowcoderofficial@sina.com
spring.mail.password=a3cf42ac35af7c17 # NOT Login password, BUT
spring.mail.protocol=smtp
spring.mail.properties.mail.smtp.ssl.enable=true # need to be set as true
```

3.utils/MailClient.java

Creating and sending MimeMessage with JavaMailSender

MimeMessagHelper simplifies the creation and configuration of MimeMessage

```
@Component
```

```
public class MailClient { // JavaMailSender -> MimeMessageHelper -> MimeMessage
    @Autowired
    private JavaMailSender mailSender;

    @Value("${spring.mail.username}")
    private String from;

    public void sendMail(String to, String subject, String content) {
        try{
            MimeMessage message = mailSender.createMimeMessage();
            MimeMessageHelper helper = new MimeMessageHelper(message); // sender helper
            helper.setFrom(from); // message, from, to, subject, content
            helper.setTo(to);
            helper.setSubject(subject);
            helper.setText(content,true);
            mailSender.send(helper.getMimeMessage());
        }catch (MessagingException e){
            logger.error("emial sent failed: "+e.getMessage());
        }
    }
}
```

4.Write test class

The error is due to: Password changed to authorization code: a3cf42ac35af7c17

❗ On June 7th, I tried registering a new user in China. No matter how many times I tested, **the activation email never arrived**.

So I ran two test functions in MailTests.java and made a surprising discovery:

> ✅ When the HTML template only contains a simple `<p>` tag, the activation email is received successfully.
> ❌ As soon as there's a URL in the HTML (like an `<a href="...">` tag), even though the function may run *successfully*, the email is not received at all.

Then I checked the error message in the console:

```
org.springframework.mail.MailAuthenticationException: Authentication failed at
org.springframework.mail.javamail.JavaMailSenderImpl.doSend(JavaMailSenderImpl.java:402)
  ...
Caused by: jakarta.mail.AuthenticationFailedException: 535 5.7.8 authentication failed
... 35 more
```

The core error is definitely related to Emailbox authentication. Possible causes include:

❓ Maybe the SMTP authorization code expired?
❓ Maybe the newly added function broke something?
❓ Maybe it's the VPN? (I noticed some VPN-related info in the logs, which made me suspicious)

After many tests, I finally discovered the real reason: **It was the VPN causing the email authentication to fail!** As soon as I turned off the VPN, the activation emails were sent and received successfully. All in all, as long as Redis is running, Kafka is running, Elasticsearch is running and VPN is OFF, then this function works perfectly. (Aaaahhhh!!! It was the VPN all along!!! I spend a whole night to find it out!

Register

Utils

```
public static String generateUUID(){
    return UUID.randomUUID().toString().replaceAll("-", ""); //只要字母和数字就够了
}
// MD5 encryption encrypts pw 1 can only encrypt but not decrypt 2 each encryption
becomes a value // MD5加密 对pw加密 1只能加密不能解密 2每次加密都是变成一个值
// hello + 3e4a8 (salt)  -> abc123def456abc
public static String md5(String key){
    if(StringUtils.isBlank(key)){
        return null;
    }
    return DigestUtils.md5DigestAsHex(key.getBytes());
}
```

Login

verification code generation

Kaptcha (https://mvnrepository.com/search?q=Kaptcha+)

1. User accesses `/kaptcha` from the frontend.
2. The server generates a random verification code (text + image).
3. The server stores the verification code text in the session, for later comparison.
4. The server sends the verification image to the browser.
5. The user sees the verification image on the login page.

The user submits the login request (`/login`).

The server retrieves the verification code text from the session and compares it with the user input:

```
session.getAttribute("kaptcha");
```

If it matches: continue checking username and password.

If it doesn't match: show "Verification code incorrect", reject login.

The `refresh_kaptcha()` function generates a new path that includes a random number `Math.random()` to prevent browser caching. `$("#kaptcha").attr("src", path);` uses jQuery selector `$("#kaptcha")` to find the `<img>` element with id="kaptcha", and sets its `src` attribute to the new path `path`, which refreshes the image.

`user_id` links the `login_ticket` table and the `user` table. `expired` is the expiration time, which decides whether the current state is logged in or logged out.

`login_ticket` is the login credential (issued to the browser).

The entire table is centered on the ticket. The ticket is the only login credential (so we know which user is logged in).

`status`: 0 = valid, 1 = invalid

```
user_id
```

`id` is auto-generated.

Data Access Layer

`entity/LoginTicket`: member variables + getter and setter + toString method

`dao/LoginTicketMapper` interface:

- use `@Mapper` annotation
- `insertLoginTicket(LoginTicket)` to insert
- `LoginTicket selectByTicket(String ticket);`
- `int updateStatus(String ticket, int status)` to update status

Method 1: `.xml` mapping

Method 2: use annotations, with a pair of parentheses containing curly braces, and multiple strings inside the braces, which form a server.

Write `testInsertLoginTicket` to test if CRUD is implemented.

Business Logic Layer

Logic: first, the input values must not be null; second...

Presentation Layer – Login

login.html:

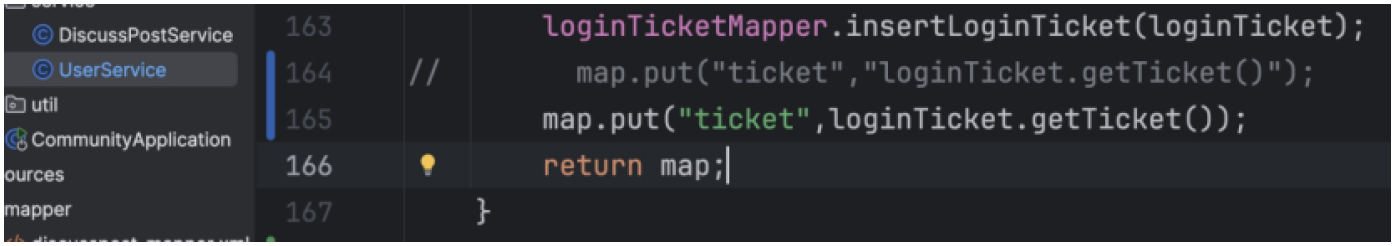
Default values

In `<input>`, `name="username"` serves to:

- specify the parameter name when submitting the form, to ensure the data is correctly sent to the server
- work with `th:value` to implement form value retention, avoiding repeated input
- used in Spring MVC for object binding, allowing form data to be mapped to Java objects automatically

Logout: the ticket status in the database stays 0 (unchanged)

Outputting the ticket: turns out the value is `loginTicket.getTicket()`. That's weird—it should be a string, and `getTicket()` shows as unused. This bug made me so frustrated.



Displaying Login Info – Using Interceptor

If logged in: header shows 123
If not logged in: header shows 456

The `user` table doesn't exist on the browser side because it's sensitive. Only the `ticket` is stored. From the ticket, we can get the `user_id`, and then we can query user info from the `user` table.

Get the cookie from the request -> wrap it into `util/CookieUtil.java

Show Login Status with Interceptor

To display the login status of the current user, we implement an Interceptor to retrieve the login ticket stored in the browser's cookie, validate it, and fetch the corresponding user. Then we make the user information accessible throughout the whole lifecycle of the request.

📌 Step 1: Create a Utility to Read Cookie Values

```
public class CookieUtil {

    public static String getValue(HttpServletRequest request, String name) {
        if(request==null||name==null){
            throw new IllegalArgumentException("The request parameter name can not be
null");
        }
        Cookie[] cookies = request.getCookies();
        if(cookies != null){
            for(Cookie cookie : cookies){
                if(cookie.getName().equals(name)){
                    return cookie.getValue();
                }
            }
        }
        return null;
    }
}
```

📌 Step 2: Use a HostHolder Utility to Replace HttpSession

HostHolder is a utility class that stores user information in a ThreadLocal, so it is accessible across the different layers of the application during a single request thread.

```
@Component
public class HostHolder {
    private ThreadLocal<User> users = new ThreadLocal<>();
    public void setUser(User user) {users.set(user);}
    public User getUser() {return users.get();}
    public void clear() {users.remove();}
}
```

📌 Step 3: Define an Interceptor

We create a LoginTicketInterceptor class that implements HandlerInterceptor and override its three core methods:
preHandle() — executed before the controller is invoked
postHandle() — executed after the controller but before the view is rendered
afterCompletion() — executed after the view is rendered (i.e., after TemplateEngine processing)

```
@Component
public class LoginTicketInterceptor implements HandlerInterceptor {

    @Autowired
    private UserService userService;

    @Autowired
    private HostHolder hostHolder;

    @Override
    public boolean preHandle(HttpServletRequest request, HttpServletResponse response,
Object handler) throws Exception {
        // Retrieve login ticket from cookie
        String ticket = CookieUtil.getValue(request, "ticket");

        if (ticket != null) {
            // Fetch login ticket from the database
            LoginTicket loginTicket = userService.findLoginTicket(ticket);
            // Check if the ticket is valid
            if (loginTicket != null && loginTicket.getStatus() == 0 &&
loginTicket.getExpired().after(new Date())) {
                // Fetch user based on the login ticket
                User user = userService.findUserById(loginTicket.getUserId());
                // Store user in current thread (for this request)
                hostHolder.setUser(user);
            }
        }
        return true;
    }

    @Override
    public void postHandle(HttpServletRequest request, HttpServletResponse response,
Object handler, ModelAndView modelAndView) throws Exception {
        User user = hostHolder.getUser();
        if (user != null && modelAndView != null) {
```

```
            // Add user to the model so that views can access it
            modelAndView.addObject("loginUser", user);
        }
    }

    @Override
    public void afterCompletion(HttpServletRequest request, HttpServletResponse response,
Object handler, Exception ex) throws Exception {
        // Clear user information to prevent memory leaks
        hostHolder.clear();
    }
}
```

📌 Step 4: Register the Interceptor in Configuration

In order that Spring can apply it to incoming requests, we need to register LoginTicketInterceptor in our WebMvcConfigurer

```
@Configuration
public class WebMvcConfig implements WebMvcConfigurer {
    @Autowired
    private LoginTicketInterceptor loginTicketInterceptor;

    @Override
    public void addInterceptors(InterceptorRegistry registry) {
        registry.addInterceptor(loginTicketInterceptor)
.excludePathPatterns("/**/*.css","/**/*.js","/**/*.png","/**/*.jpg","/**/*.jpeg");
    }
}
```

📌 Step 5: index.html

Use the `loginUser` object, which is stored in the Model, to check whether the user is logged in.
If the user is **logged in**, display **"Messages"** and **"My Profile"** on the homepage.
If the user is **not logged in**, display **"Sign up"** and **"Sign in"**.

Messages
Sign up | Sign in

User Avatar

Posts Modul

filter sensitive words with Prefix Tree(Trie)

WHAT is Trie?

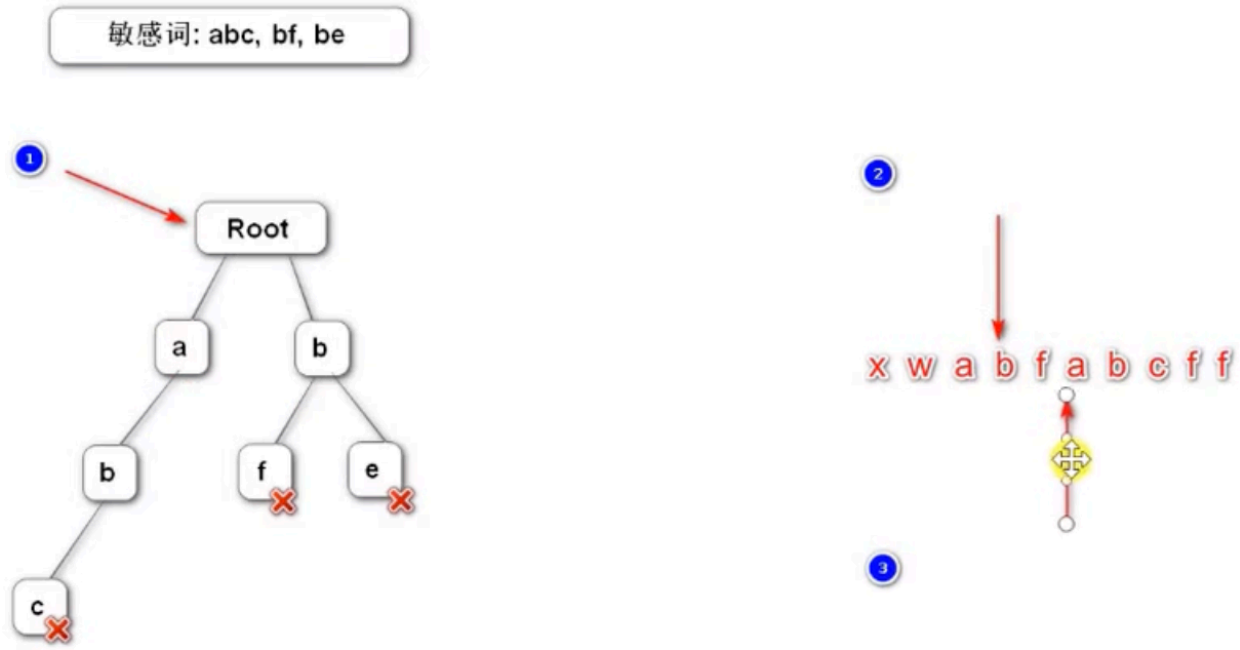
A Trie is a tree data structure used to store strings efficiently. The root node contains no character. Every child node (except the root) contains exactly one character. A path from the root to a leaf forms a string. If the string is a sensitive word, the leaf node is marked. Among sibling nodes, each character is unique.

Trie is trading memory space for time.

Applications: String searching, Word frequency statistics(HOT words), String sorting

You use a `StringBuilder`:

- If the substring is **not** a sensitive word, append the characters as they are.
- If it **is** a sensitive word, replace the entire word with `*` symbols.
- After replacement, move Pointer 2 forward and reset Pointer 1 to the root.
- input is a text, output is a text without sensitive words



There are three pointers**:** Pointer 1: Points to the root of the Trie, used to traverse the tree. Pointer 2: Points to the current character in the string from left to right; marks the start of a potential word. It never moves backward. Pointer 3: Starts from Pointer 2 and moves forward; marks the end of a potential word. It resets when Pointer 2 moves forward.

HOW?

Step1: Define the prefix tree. util/class SensitiveFilter

Step2: Initialize it using the list of sensitive words. resources/sensitive-words.txt

Step3: Implement a filtering method that checks and replaces sensitive words. public String filter(String text){

post with AJAX

AJAX stands for Asynchronous JavaScript and XML. It's not a new technology, but rather a new way of using existing technologies. With AJAX, a web page can update parts of its content asynchronously — that is, without refreshing the entire page. Although the "X" stands for XML, nowadays JSON is more commonly used because it's easier to parse and work with in JavaScript. 增量更新显示在页面上，不需要刷新页面

Using **jQuery** to send an AJAX request: In an HTML page, we can add a small button. When the button is clicked, a **POST** request is sent asynchronously. This allows us to interact with the server and retrieve data **without reloading the page**.

❓ Two Ways to send user input data from Frontend to Backend in a Spring Boot web app:

1. **Form Submission** (e.g., login, register):
    - Sends data by reloading the page.
    - Spring MVC automatically binds form fields to Java objects using matching `name` attributes.
2. **AJAX Request** (e.g., posting, commenting):
    - Sends data asynchronously without reloading the page.
    - You manually collect input and send it using `$.post()` or `fetch()`.
    - The backend receives it as individual parameters or a JSON object.

👉 **Use form submission for page navigation; use AJAX for smoother user experience with partial updates.**

`loginUser` is a user object passed from the backend controller to the HTML page via the Model. The `th:if="${loginUser!=null}"` condition in Thymeleaf ensures that the Post button is only displayed if a user is logged in.

```
th:if="${loginUser!=null}"
// if logged in, then show "Post"
```

❓ After click a post, what happened?

Click -> changen url(add postId after url)

turn to `discuss/detail` site

Spring MVC's automatic binding mechanism: The `name` attribute in the HTML form must strictly match the field name of the Java class (case-sensitive) and must be accessed through Getter/Setter: It is required to access properties through getter/setter (even if the field is `private`), not directly .name, which is a manifestation of encapsulation. Spring MVC 的自动绑定机制：HTML 表单中的 `name` 属性必须与 Java 类的字段名 **严格匹配**（区分大小写）**必须通过 Getter/Setter 访问**：要求通过 getter/setter 访问属性（即使字段是 `private`），不能直接.name，这是封装性的体现。

```
    <select id="selectDiscussPosts" resultType="DiscussPost">
        select <include refid="selectFields"></include>
        from discuss_post
        where status != 2
--          如果传进来的 userId 不等于 0，就加上 and user_id = #{userId} 这个过滤条件。
        <if test="userId!=0">
            and user_id = #{userId}
        </if>
```

```
        <if test="orderMode==0">
            order by type desc, create_time desc
        </if>
        <if test="orderMode==1">
            order by type desc, score desc, create_time desc
        </if>
        limit #{offset}, #{limit}
    </select>
```

Spring Data Access - Transaction Management

The phantom read condition is to count multiple data (such as the total number of posts), which is acceptable, for example, we count in the middle of the night 幻读条件是统计多条数据（比如帖子总数），可以接受，比如我们后半夜统计

Optimistic lock: Even if there is concurrency, there is usually no problem 乐观锁：即使并发了，通常也不会有问题

Declarative Transaction Management

Programmatic Transaction Management: only want to control a part of it.

Show Comments in Post Detail Page

entity_type: type(target of comment) 1:post, 2:comment

entity_id: id of that post/comment

target_id: Under a post, you can also comment a comment, whom you comment to

Status: 2:deleted

- `entity_type` + `entity_id` 共同构成 **复合外键**，实现评论与多种实体关联的灵活性。

  ```
  where status=0
  and entity_type=#{entityType}
  and entity_id=#{entityId}
  ```

Add a Comment

Add a Comment in db, then revise the number of comments

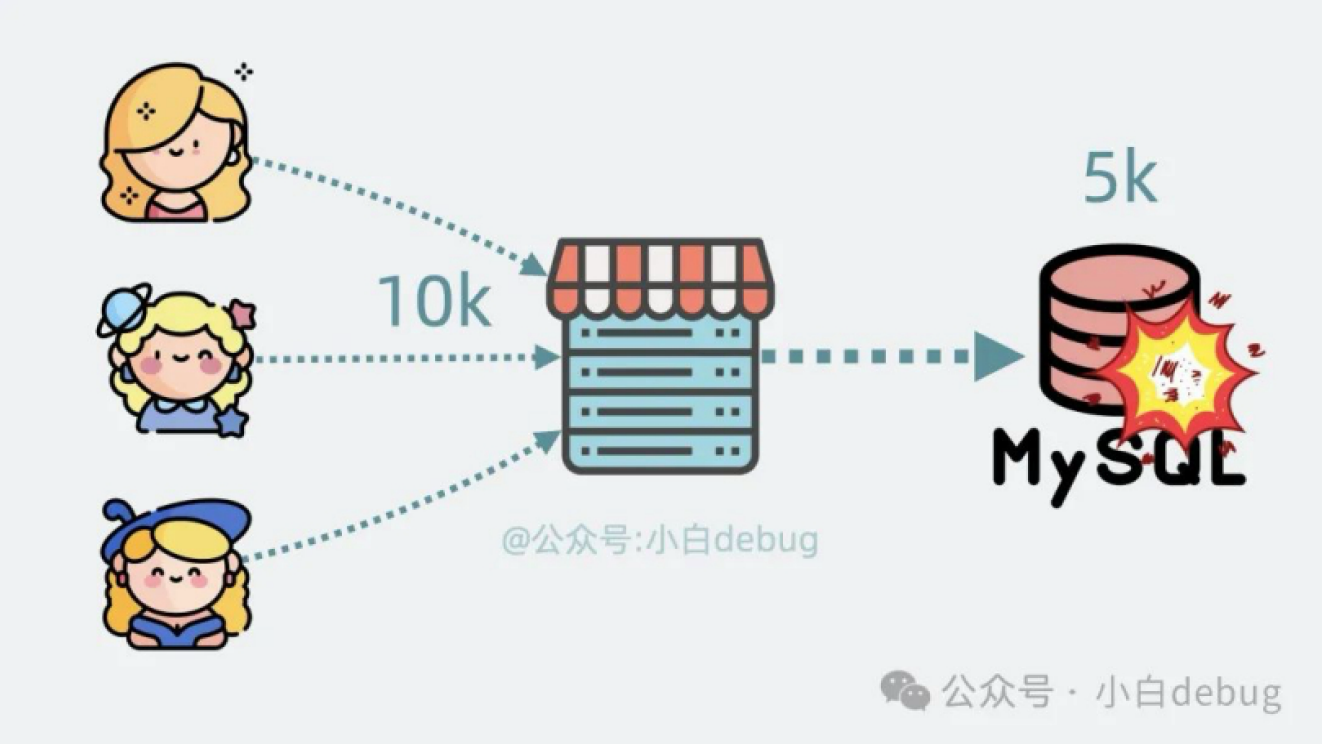
Messages

Redis - like

A lightning-fast, in-memory cache & database for high-speed data access. Redis解决QPS高的问题，比如双11秒杀或者春运抢车票，Redis是内存存储 + 单线程IO多路复用 → 轻松支撑10万+ QPS。提升的是数据查询性能, 避免MySQL被突发流量击穿

As a programmer maintaining a **Product Service**:

- Service needs to handle **10,000 QPS** (queries per second)
- Underlying MySQL can only support **5,000 QPS**
- eg. Double 11 flash sales, Spring Festival train ticket rush, Any high-concurrency read operations

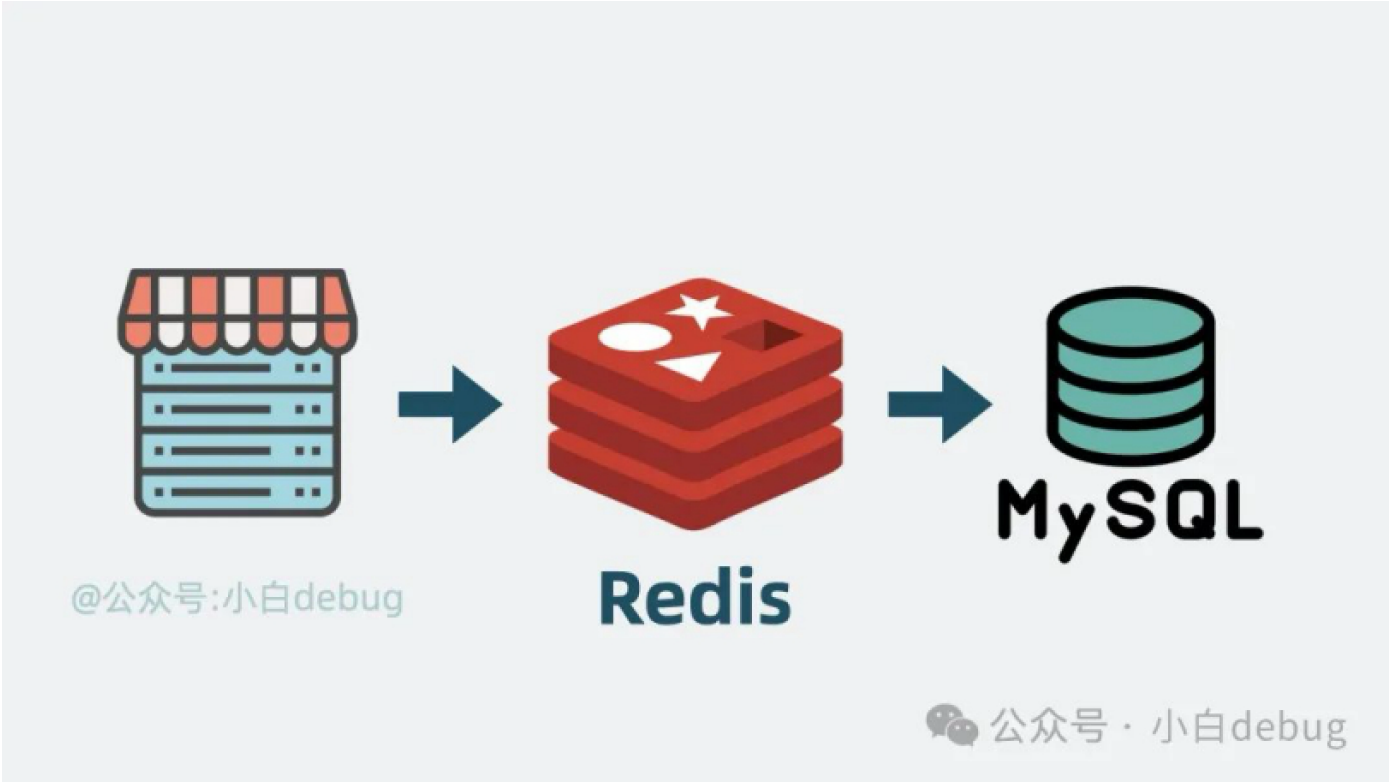


Key insight:

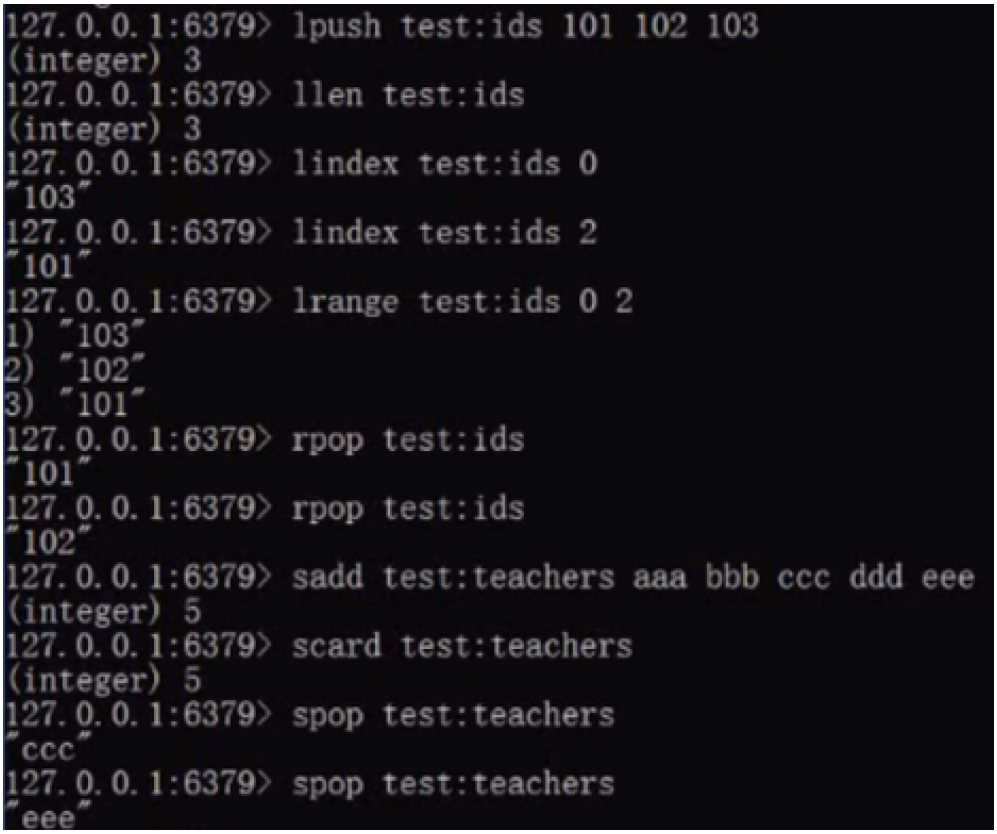
- Memory access is **100x faster** than disk access
- MySQL stores data on disk → Slow
- Redis stores data in memory → Fast

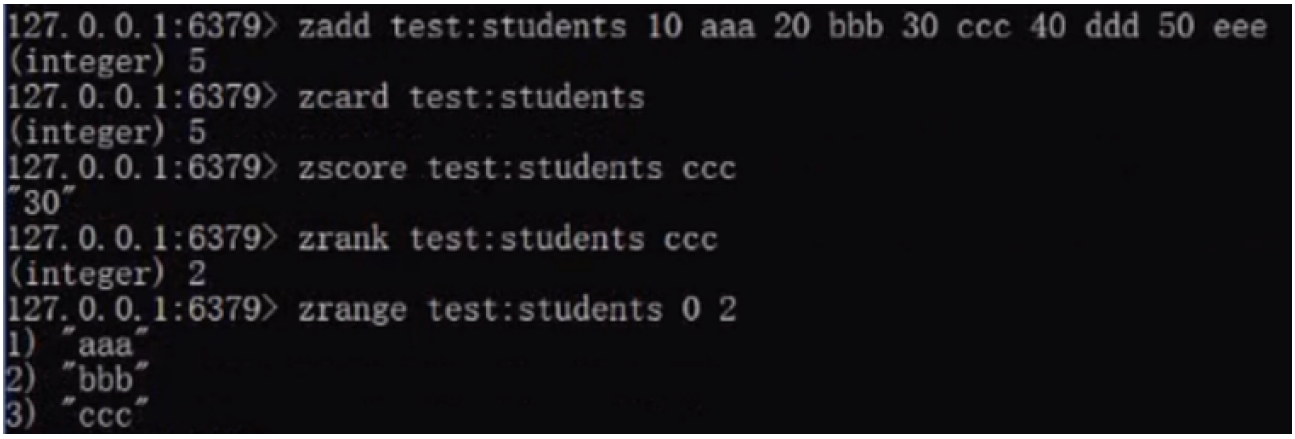
Implementation approach:

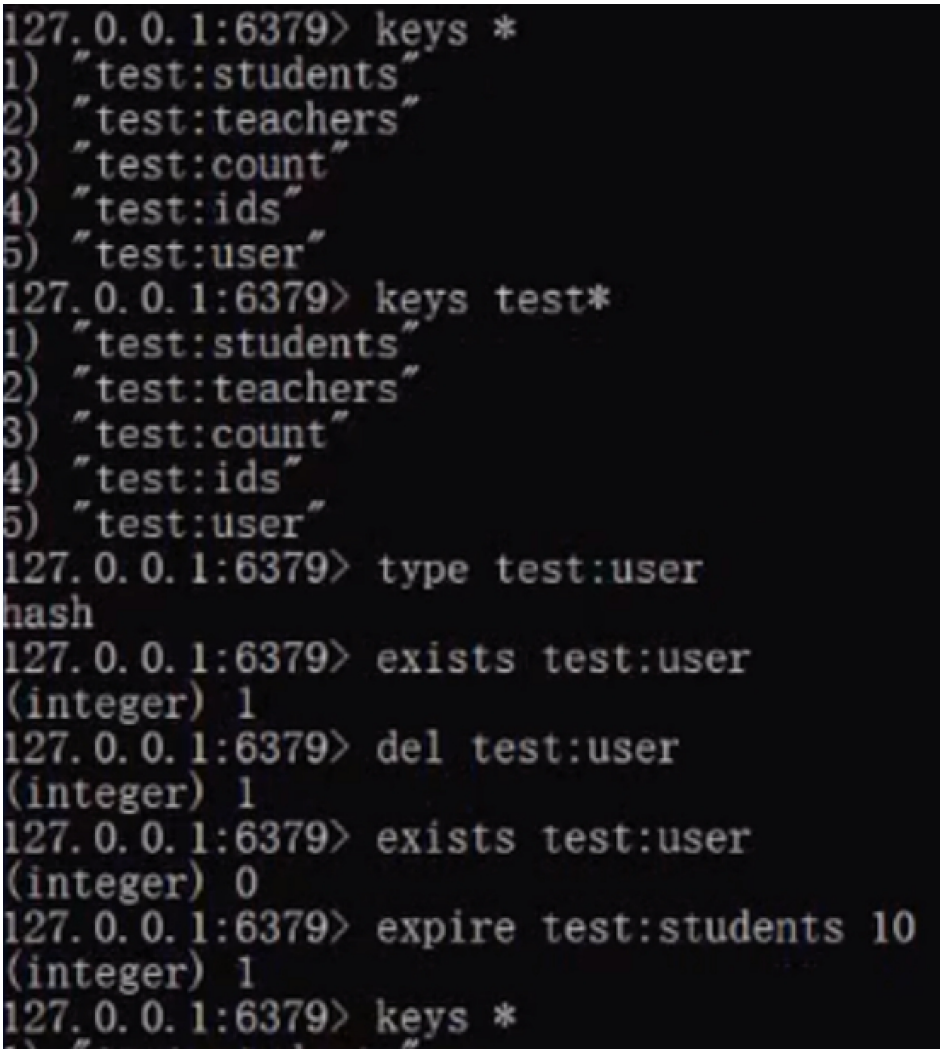
1. Check Redis (in-memory) first
2. If missing ("cache miss", which means when data isn't found in Redis and must be fetched from MySQL), query MySQL
3. Store MySQL results in Redis for future requests




```
C:\Users\Administrator>redis-cli
127.0.0.1:6379> select 1
OK
127.0.0.1:6379[1]> select 2
OK
127.0.0.1:6379[2]> select 0
OK
127.0.0.1:6379> flushdb
OK
127.0.0.1:6379> set test:count 1
OK
127.0.0.1:6379> get test:count
"1"
127.0.0.1:6379> incr test:count
(integer) 2
127.0.0.1:6379> decr test:count
(integer) 1
127.0.0.1:6379> hset test:user id 1
(integer) 1
127.0.0.1:6379> hset test:user username zhangsan
(integer) 1
127.0.0.1:6379> hget test:user id
"1"
127.0.0.1:6379> hget test:user username
"zhangsan"
```


127.0.0.1:6379> lpush test:ids 101 102 103
(integer) 3
127.0.0.1:6379> llen test:ids
(integer) 3
127.0.0.1:6379> lindex test:ids 0
"103"
127.0.0.1:6379> lindex test:ids 2
"101"
127.0.0.1:6379> lrange test:ids 0 2
1) "103"
2) "102"
3) "101"
127.0.0.1:6379> rpop test:ids
"101"
127.0.0.1:6379> rpop test:ids
"102"
127.0.0.1:6379> sadd test:teachers aaa bbb ccc ddd eee
(integer) 5
127.0.0.1:6379> scard test:teachers
(integer) 5
127.0.0.1:6379> spop test:teachers
"ccc"
127.0.0.1:6379> spop test:teachers
"eee"


127.0.0.1:6379> zadd test:students 10 aaa 20 bbb 30 ccc 40 ddd 50 eee
(integer) 5
127.0.0.1:6379> zcard test:students
(integer) 5
127.0.0.1:6379> zscore test:students ccc
"30"
127.0.0.1:6379> zrank test:students ccc
(integer) 2
127.0.0.1:6379> zrange test:students 0 2
1) "aaa"
2) "bbb"
3) "ccc"



spring redis

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-redis</artifactId>
    <version>3.4.3</version>
</dependency>
```

```
# redis
spring.data.redis.database=11
spring.data.redis.host=localhost
spring.data.redis.port=6379
```

```
@Autowired
private RedisTemplate redisTemplate;
```

👉最终 Redis 中的数据是：

- `like:entity:1:233` → Set：[12, 44, 88] 表示这条 entity 被这几个用户点过赞
- `like:user:44` → 109 表示用户 44 被赞了 109 次

Redis的事务管理

在 Spring 的 Redis 中，事务管理是通过 `RedisTemplate.execute(new SessionCallback(){...})` + `operations.multi()` + `operations.exec()` 来手动控制的，而不是像数据库那样使用 `@Transactional` 注解。

Follow

📣 Kafka-Based System Notifications (e.g. “Someone commented on your post”)

When you **comment**, **like**, or **follow**, the system automatically sends a notification such as:

> “Alice commented on your post”
> and includes a link to that post or user.

These user interactions happen **frequently**, so to **improve performance**, we use **Kafka** as a message queue to process them **asynchronously and concurrently**.
Kafka is a **distributed message queue** system. Even without Kafka, you could use a simple **blocking queue**, but Kafka has key advantages:

- Kafka **persists messages to disk**, while blocking queues usually store them in memory.
- **Disk is cheaper and larger than memory**, allowing Kafka to handle large volumes of data.
- Kafka reads/writes are **sequential and fast**, making it ideal for logging, messaging, and event processing.

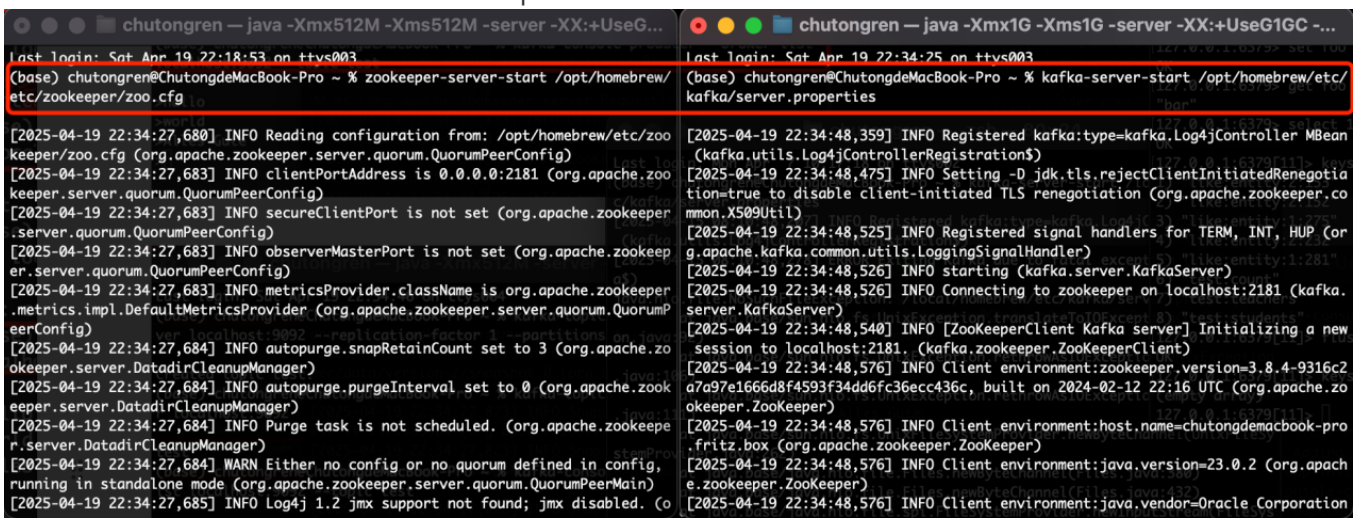
🧠 Kafka Terminology

- Broker: A Kafka server that stores and serves messages.
- Zookeeper: Manages Kafka brokers and handles cluster coordination.
- Topic: Like a folder, used to store messages of a specific category.
- Partition: A subdivision of a topic to allow parallelism and scalability.

⚙️ Kafka Installation

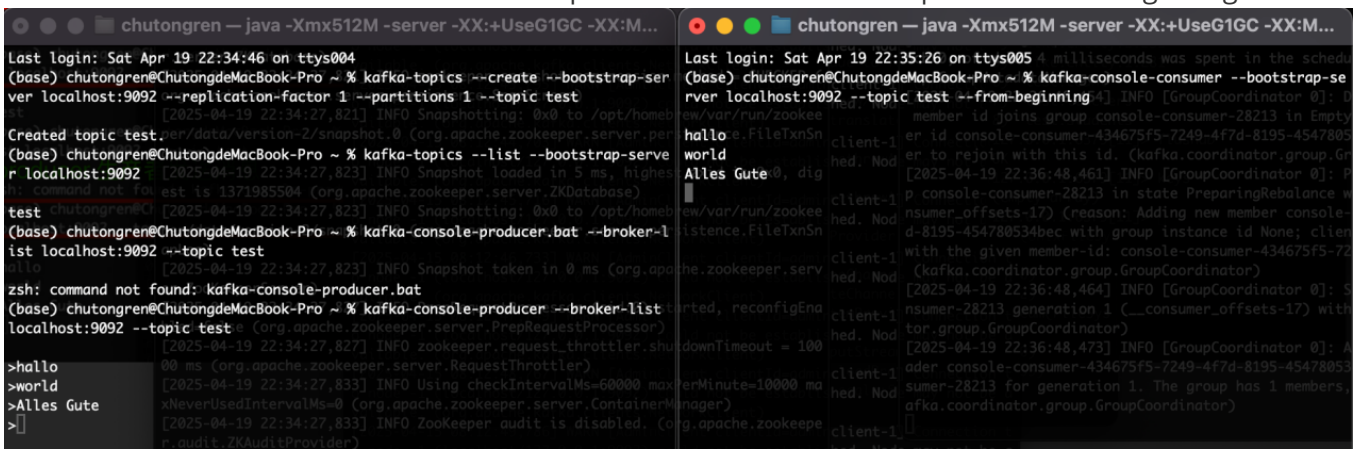
step1: Install Kafka with homebrew
brew install kafka
step2: Configuration (optional)
/opt/homebrew/etc/kafka/server.properties
don’t need to change anything for basic use — default paths will work.
step3: start services
open two separate terminal tabs:

zookeeper-server-start /opt/homebrew/etc/zookeeper/zoo.cfg
kafka-server-start /opt/homebrew/etc/kafka/server.properties
ATTENTION: Make sure Zookeeper is started before Kafka!



step 4: create a kafka topic
kafka-topics.bat --create --bootstrap-server localhost:9002 --replication 1 --partition --topic test
kafka-topics.bat --list --bootstrap-server localhost:9002

step 5: producer and consumer send and consume messages
kafka-console-producer --broker-list localhost:9092 --topic test
kafka-console-consumer --bootstrap-server localhost:9092 --topic test --from-beginning



Spring Boot Integration with Kafka

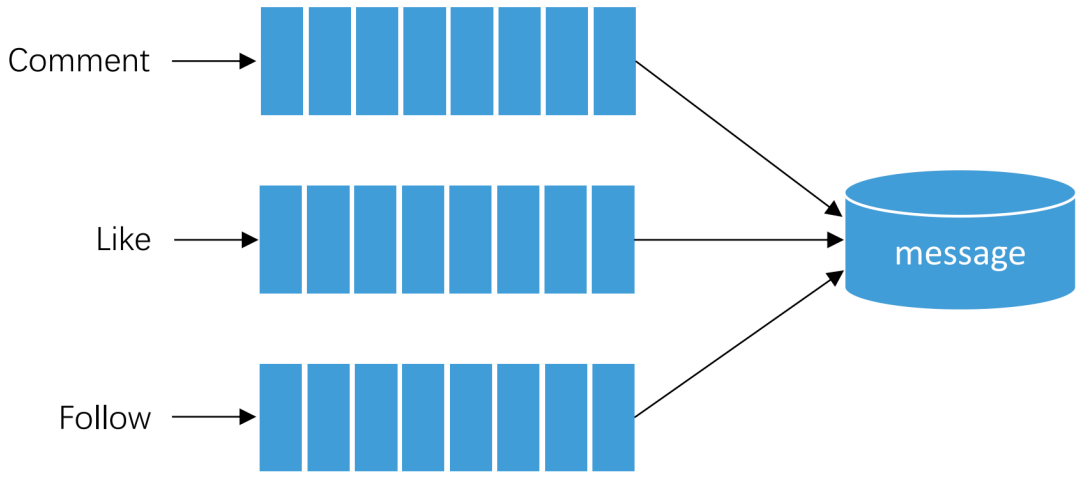
spring kafka

```
# KafkaProperties
spring.kafka.bootstrap-servers=localhost:9092
spring.kafka.consumer.group-id=test-consumer-group
spring.kafka.consumer.enable-auto-commit=true
spring.kafka.consumer.auto-commit-interval=3000 # ms
```

KafkaProducer -> `KafkaTemplate.send(topic,data)` this is a active call

`@KafkaListener(topics = {"test"}), public void handleMessage(ConsumerRecord record)` Passive trigger, automatically as long as there is an event in the queue

Send System Notifications



entity/Event

```
public class Event {
    private String topic; // "comment", "like" or "follow"
    private int userId; // always = 1
    private int entityType;
    private int entityId;
    private int entityUserId;
    private Map<String, Object> data = new HashMap<>(); // link
}
```

event/EventProducer, EventConsumer event data transfer with JSON Format

```
@Component
public class EventProducer {

    @Autowired
    private KafkaTemplate kafkaTemplate;

    public void fireEvent(Event event) {
        kafkaTemplate.send(event.getTopic(), JSONObject.toJSONString(event));
    }
}
```

```
@Component
public class EventConsumer implements CommunityConstant {

  @Autowired
  private MessageService messageService;

  @KafkaListener(topics = {TOPIC_COMMENT, TOPIC_LIKE, TOPIC_FOLLOW})
  public void handleCommentMessage(ConsumerRecord record) {
      if (record == null || record.value() == null) {
          logger.error("The kafka message content is null!");
          return;
      }
```

```
        Event event = JSONObject.parseObject(record.value().toString(), Event.class);
        if (event == null) {
            logger.error("Invalid kafka message format!");
            return;
        }

        Message message = new Message();
        message.setFromId(SYSTEM_USER_ID);
        message.setToId(event.getEntityUserId());
        message.setConversationId(event.getTopic());
        message.setCreateTime(new Date());

        Map<String, Object> content = new HashMap<>();
        content.put("userId", event.getUserId());
        content.put("entityType", event.getEntityType());
        content.put("entityId", event.getEntityId());

        if (!event.getData().isEmpty()) {
            for (Map.Entry<String, Object> entry : event.getData().entrySet()) {
                content.put(entry.getKey(), entry.getValue());
            }
        }

        message.setContent(JSONObject.toJSONString(content));
        messageService.addMessage(message);
    }
}
```

controller/　把Event准备好给eventProducer

```
@Controller
@RequestMapping("/comment")
public class CommentController implements CommunityConstant {
  @RequestMapping(path = "/add/{discussPostId}", method = RequestMethod.POST)
  public String addComment(@PathVariable("discussPostId") int discussPostId, Comment
comment) {
        // ...

        Event event = new Event()
                .setTopic(TOPIC_COMMENT)
                .setUserId(hostHolder.getUser().getId())
                .setEntityType(comment.getEntityType())
                .setEntityId(comment.getEntityId())
                .setData("postId", discussPostId);
        if (comment.getEntityType() == ENTITY_TYPE_POST) {
            DiscussPost target =
discussPostService.findDiscussPostById(comment.getEntityId());
            event.setEntityUserId(target.getUserId());
        } else if (comment.getEntityType() == ENTITY_TYPE_COMMENT) {
            Comment target = commentService.findCommentById(comment.getEntityId());
            event.setEntityUserId(target.getUserId());
```

```
        }
        eventProducer.fireEvent(event);

        return "redirect:/discuss/detail/" + discussPostId;
    }
}
```

```
@Controller
public class LikeController implements CommunityConstant {
    @RequestMapping(path = "/like", method = RequestMethod.POST)
    @ResponseBody
    public String like(int entityType, int entityId, int entityUserId, int postId) {
        // ...

        if (likeStatus == 1) {
            Event event = new Event()
                    .setTopic(TOPIC_LIKE)
                    .setUserId(hostHolder.getUser().getId())
                    .setEntityType(entityType)
                    .setEntityId(entityId)
                    .setEntityUserId(entityUserId)
                    .setData("postId", postId);
            eventProducer.fireEvent(event);
        }

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

```
@Controller
public class FollowController implements CommunityConstant {
    @RequestMapping(path = "/follow", method = RequestMethod.POST)
    @ResponseBody
    public String follow(int entityType, int entityId) {
        User user = hostHolder.getUser();
        followService.follow(user.getId(), entityType, entityId);

        Event event = new Event()
                .setTopic(TOPIC_FOLLOW)
                .setUserId(hostHolder.getUser().getId())
                .setEntityType(entityType)
                .setEntityId(entityId)
                .setEntityUserId(entityId);
        eventProducer.fireEvent(event);

        return CommunityUtil.getJSONString(0, "Followed!");
    }
}
```

Show unread numbers

```
Message selectLatestNotice(int userId, String topic);

int selectNoticeCount(int userId, String topic);

int selectNoticeUnreadCount(int userId, String topic);

List<Message> selectNotices(int userId, String topic, int offset, int limit);
```

```
<select id="selectLatestNotice" resultType="Message">
    select <include refid="selectFields"></include>
    from message
    where id in (
    select max(id) from message
    where status != 2
    and from_id = 1
    and to_id = #{userId}
    and conversation_id = #{topic}
    )
</select>

<select id="selectNoticeCount" resultType="int">
    select count(id) from message
    where status != 2
    and from_id = 1
    and to_id = #{userId}
      and conversation_id = #{topic}
</select>

<select id="selectNoticeUnreadCount" resultType="int">
    select count(id) from message
    where status = 0
    and from_id = 1
    and to_id = #{userId}
    <if test="topic!=null">
        and conversation_id = #{topic}
    </if>
</select>

<select id="selectNotices" resultType="Message">
    select <include refid="selectFields"></include>
    from message
    where status != 2
    and from_id = 1
    and to_id = #{userId}
    and conversation_id = #{topic}
    order by create_time desc
    limit #{offset}, #{limit}
</select>
```

```
public Message findLatestNotice(int userId, String topic) {
```

```
    return messageMapper.selectLatestNotice(userId, topic);
}

public int findNoticeCount(int userId, String topic) {
    return messageMapper.selectNoticeCount(userId, topic);
}

public int findNoticeUnreadCount(int userId, String topic) {
    return messageMapper.selectNoticeUnreadCount(userId, topic);
}

public List<Message> findNotices(int userId, String topic, int offset, int limit) {
    return messageMapper.selectNotices(userId, topic, offset, limit);
}
```

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

    Message message = messageService.findLatestNotice(user.getId(), TOPIC_COMMENT);
    Map<String, Object> messageVO = new HashMap<>();
    if (message != null) {
        messageVO.put("message", message);

        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"));

        int count = messageService.findNoticeCount(user.getId(), TOPIC_COMMENT);
        messageVO.put("count", count);

        int unread = messageService.findNoticeUnreadCount(user.getId(), TOPIC_COMMENT);
        messageVO.put("unread", unread);
    }
    model.addAttribute("commentNotice", messageVO);


    message = messageService.findLatestNotice(user.getId(), TOPIC_LIKE);
    messageVO = new HashMap<>();
    if (message != null) {
        messageVO.put("message", message);

        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"));

        int count = messageService.findNoticeCount(user.getId(), TOPIC_LIKE);
        messageVO.put("count", count);

        int unread = messageService.findNoticeUnreadCount(user.getId(), TOPIC_LIKE);
        messageVO.put("unread", unread);
    }
    model.addAttribute("likeNotice", messageVO);


    message = messageService.findLatestNotice(user.getId(), TOPIC_FOLLOW);
    messageVO = new HashMap<>();
    if (message != null) {
        messageVO.put("message", message);

        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"));

        int count = messageService.findNoticeCount(user.getId(), TOPIC_FOLLOW);
        messageVO.put("count", count);

        int unread = messageService.findNoticeUnreadCount(user.getId(), TOPIC_FOLLOW);
        messageVO.put("unread", unread);
    }
    model.addAttribute("followNotice", messageVO);

    // count(unread)
    int letterUnreadCount = messageService.findLetterUnreadCount(user.getId(), null);
    model.addAttribute("letterUnreadCount", letterUnreadCount);
    int noticeUnreadCount = messageService.findNoticeUnreadCount(user.getId(), null);
    model.addAttribute("noticeUnreadCount", noticeUnreadCount);

    return "/site/notice";
}

@RequestMapping(path = "/notice/detail/{topic}", method = RequestMethod.GET)
public String getNoticeDetail(@PathVariable("topic") String topic, Page page, Model
model) {
    User user = hostHolder.getUser();

    page.setLimit(5);
    page.setPath("/notice/detail/" + topic);
    page.setRows(messageService.findNoticeCount(user.getId(), topic));

    List<Message> noticeList = messageService.findNotices(user.getId(), topic,
page.getOffset(), page.getLimit());
```

```
    List<Map<String, Object>> noticeVoList = new ArrayList<>();
    if (noticeList != null) {
        for (Message notice : noticeList) {
            Map<String, Object> map = new HashMap<>();
            // 通知
            map.put("notice", notice);
            // 内容
            String content = HtmlUtils.htmlUnescape(notice.getContent());
            Map<String, Object> data = JSONObject.parseObject(content, HashMap.class);
            map.put("user", userService.findUserById((Integer) data.get("userId")));
            map.put("entityType", data.get("entityType"));
            map.put("entityId", data.get("entityId"));
            map.put("postId", data.get("postId"));
            // 通知作者
            map.put("fromUser", userService.findUserById(notice.getFromId()));

            noticeVoList.add(map);
        }
    }
    model.addAttribute("notices", noticeVoList);

    // 设置已读
    List<Integer> ids = getLetterIds(noticeList);
    if (!ids.isEmpty()) {
        messageService.readMessage(ids);
    }

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

write a postman function

Elasticsearch Fundamentals

1. Introduction to Elasticsearch

- A **distributed**, **RESTful** search and analytics engine
- Capabilities:
  - Supports **full-text search** across diverse data types
  - Delivers **real-time** search results with high speed
  - Designed for **horizontal scaling** (handles PB-scale data per second)

2. Core Terminology

Term	Definition
Index 索引（db）-> Table	A collection of documents (analogous to a database in SQL)
Type 类型（Table 一张表）	~~Deprecated in v7.0+~~ (Formerly a logical category within an index)
Document 文档（行，一条数据）	Basic unit of data (JSON format), like a "row" in SQL
Field 字段（列）	A key-value pair within a document (e.g., `"title": "Elasticsearch 101"`)
Cluster 集群（一群服务器）	A set of connected nodes that hold all data
Node 节点（其中一台）	A single server in the cluster
Shard 分片（一个索引拆分成多个分片去存，提高并发能力）	A subset of an index's data (enables distributed storage)
Replica 副本（提高可用性）	A copy of a shard for fault tolerance

Setting Up Elasticsearch for Search: A Step-by-Step Guide

1. Overview

- **Purpose**: Sync database data to Elasticsearch (ES) for fast full-text search (default port: 9200)
- **Prerequisite Services**:

```
brew services start redis
zookeeper-server-start /opt/homebrew/etc/zookeeper/zoo.cfg
kafka-server-start /opt/homebrew/etc/kafka/server.properties
./bin/elasticsearch  # Launch ES
```

---

2. Installation & Configuration

A. Download & Run

1. Download from Elasticsearch Official Site
2. Navigate to the installation folder and start:

```
cd /Users/chutongren/Documents/job/nowcoder/tools/elasticsearch-7.17.1
./bin/elasticsearch
```

B. Fixing macOS Security Errors

- **Issue**: macOS blocks unsigned JDK on Apple Silicon (M1/M2/M3)
- **Solution**:

```
sudo xattr -rd com.apple.quarantine /Library/Java/JavaVirtualMachines/jdk-23.jdk
```

C. Install Chinese IK Analyzer Plugin

- **Method 1 (CLI)**:

```
bin/elasticsearch-plugin install https://get.infini.cloud/elasticsearch/analysis-ik/7.17.1
```

- **Method 2 (Manual)**:
  1. Download plugin ZIP from IK GitHub
  2. Unzip to `plugins/ik/` in the ES installation directory
- **Custom Words**: Configure new terms in `plugins/ik/config/IKAnalyzer.cfg.xml`

D. Configure `elasticsearch.yml`

```
cluster.name: nowcoder
path.data: /Users/chutongren/Documents/job/nowcoder/tools/elasticsearch-7.17.1/data
path.logs: /Users/chutongren/Documents/job/nowcoder/tools/elasticsearch-7.17.1/logs
// xpack.security.enabled: false  # Disable auth for local dev
```

3. Integration with Spring Boot

- **Application Properties**:

```
spring.elasticsearch.uris=localhost:9200  # No "http://" needed
```

- **Conflict Note**: ES 7+ no longer clashes with Netty (used by Redis).

---

4. Verification & Troubleshooting

A. Check ES Health

```
curl -X GET "localhost:9200/_cat/health?v"
```

- **Error Fix**: If connection fails, clear SSL passwords:

```
./bin/elasticsearch-keystore remove
xpack.security.transport.ssl.keystore.secure_password
./bin/elasticsearch-keystore remove xpack.security.http.ssl.keystore.secure_password
```

- **Test**: Access `http://localhost:9200` in browser or via `curl`.

B. Postman/curl Examples

- Create an index:

```
curl -X PUT "localhost:9200/my_index"
```

- Search data:

```
curl -X GET "localhost:9200/my_index/_search?q=title:elasticsearch"
```

---

5. Key Takeaways

- **ES vs. DB**: ES is a search engine, not a primary database. Sync only search-needed fields.
- **Performance**: Avoid wildcard queries (`*text*`) for better speed.
- **Security**: Disable `xpack.security` **only for local development**.

Here's a polished English version of your Elasticsearch implementation notes, structured for a technical blog post with clear explanations and code samples:

Implementing Elasticsearch Search in Spring Boot: Version Compatibility Solutions

1. Version Downgrade Rationale

- **Problem**: Code tutorials used Elasticsearch 7.x, but initial setup with ES 9.0.0 failed due to API changes.
- **Solution**: Switched to **ES 7.17.10** (LTS version) with tested code from this guide.

---

2. Key Implementation Steps

A. Entity Mapping (`DiscussPost.java`)

```
import org.springframework.data.elasticsearch.annotations.*;

@Document(indexName = "discusspost")  // ES index name
public class DiscussPost {
    @Id
    private int id;

    @Field(type = FieldType.Text, analyzer = "ik_max_word")  // Chinese分词
    private String title;

    @Field(type = FieldType.Text, analyzer = "ik_max_word")
    private String content;

    @Field(type = FieldType.Integer)
    private int type;  // 1=置顶, 0=普通

    @Field(type = FieldType.Double)
    private double score;  // 权重

    @Field(type = FieldType.Date, format = DateFormat.basic_date_time)
    private Date createTime;
}
```

B. Repository Interface (Unmodified)

```
public interface DiscussPostRepository extends ElasticsearchRepository<DiscussPost,
Integer> {
    // Inherits CRUD methods like save(), delete(), search()
}
```

---

3. Search Implementation (Test Class)

```
import org.springframework.data.elasticsearch.core.*;
import org.springframework.data.elasticsearch.core.query.*;

public class ElasticsearchTests {

    @Autowired
    private ElasticsearchOperations elasticsearchOperations;
```

```
    @Autowired
    private DiscussPostRepository discussRepository;

    // 1. Insert Test Data
    @Test
    public void testInsert() {
        DiscussPost post = new DiscussPost();
        post.setId(1001);
        post.setTitle("互联网寒冬求职指南");
        post.setContent("大厂裁员潮下的职业规划...");
        post.setType(0);
        post.setScore(88.8);
        post.setCreateTime(new Date());
        discussRepository.save(post);
    }

    // 2. Complex Search
    @Test
    public void testSearch() {
        // Step 1: Build Query
        NativeSearchQuery query = new NativeSearchQueryBuilder()
            .withQuery(QueryBuilders.multiMatchQuery("互联网寒冬", "title", "content"))
            .withSort(SortBuilders.fieldSort("type").order(SortOrder.DESC))      // 置顶优
先
            .withSort(SortBuilders.fieldSort("score").order(SortOrder.DESC))    // 高分优先
            .withSort(SortBuilders.fieldSort("createTime").order(SortOrder.DESC))
            .withPageable(PageRequest.of(0, 10))  // 分页
            .withHighlightFields(
                new HighlightBuilder.Field("title").preTags("<em>").postTags("</em>"),
                new HighlightBuilder.Field("content").preTags("<em>").postTags("</em>")
            )
            .build();

        // Step 2: Execute
        SearchHits<DiscussPost> hits = elasticsearchOperations.search(query,
DiscussPost.class);

        // Step 3: Process Results
        hits.getSearchHits().forEach(hit -> {
            DiscussPost post = hit.getContent();
            Map<String, List<String>> highlights = hit.getHighlightFields();
            // Apply highlights (if any)
            if (highlights.containsKey("title")) {
                post.setTitle(highlights.get("title").get(0));
            }
            System.out.println(post);
        });
    }
}
```

4. Critical Notes

A. API Compatibility

- **Avoid deprecated clients**:
  - ES 5.x: `TransportClient` ❌
  - ES 7.x: `RestHighLevelClient` ❌
  - **Recommended**:
    - Spring Data Elasticsearch (as shown above)
    - Or ES 8+ Java Client (Official Docs)

B. Highlighting Workaround

- ES returns raw data with `<em>` tags – frontend handles rendering:

```
// React/Vue example
<div v-html="post.title"></div>  // Renders <em> tags as red
```

C. Why Downgrade Worked

- Spring Data Elasticsearch 4.x (used here) has **best compatibility** with ES 7.x.
- ES 9.x requires Spring Data Elasticsearch 5.x + major code refactoring.

---

5. Verification Steps

1. **Check indices**:

```
curl -X GET "localhost:9200/_cat/indices?v"
```

2. **Query data**:

```
curl -X GET "localhost:9200/discusspost/_search?q=title:互联网寒冬"
```

GET localhost:9200/discusspost/_mapping Send

Params Authorization Headers (7) Body Scripts Tests Settings Cookies

Body Cookies Headers (5) Test Results 200 OK 47 ms 715 B Save Response

JSON Preview Visualize


```
{
    "discusspost": {
        "mappings": {
            "properties": {
                "_class": {
                    "type": "keyword",
                    "index": false,
                    "doc_values": false
                },
                "commentCount": {
                    "type": "integer"
                },
                "content": {
                    "type": "text",
                    "analyzer": "ik_max_word",
                    "search_analyzer": "ik_smart"
                },
                "createTime": {
                    "type": "date",
                    "format": "date_optional_time||epoch_millis"
                },
                "id": {
                    "type": "long"
                },
                "score": {
```


GET localhost:9200/_analyze Send

Params Authorization Headers (9) Body Scripts Tests Settings Cookies

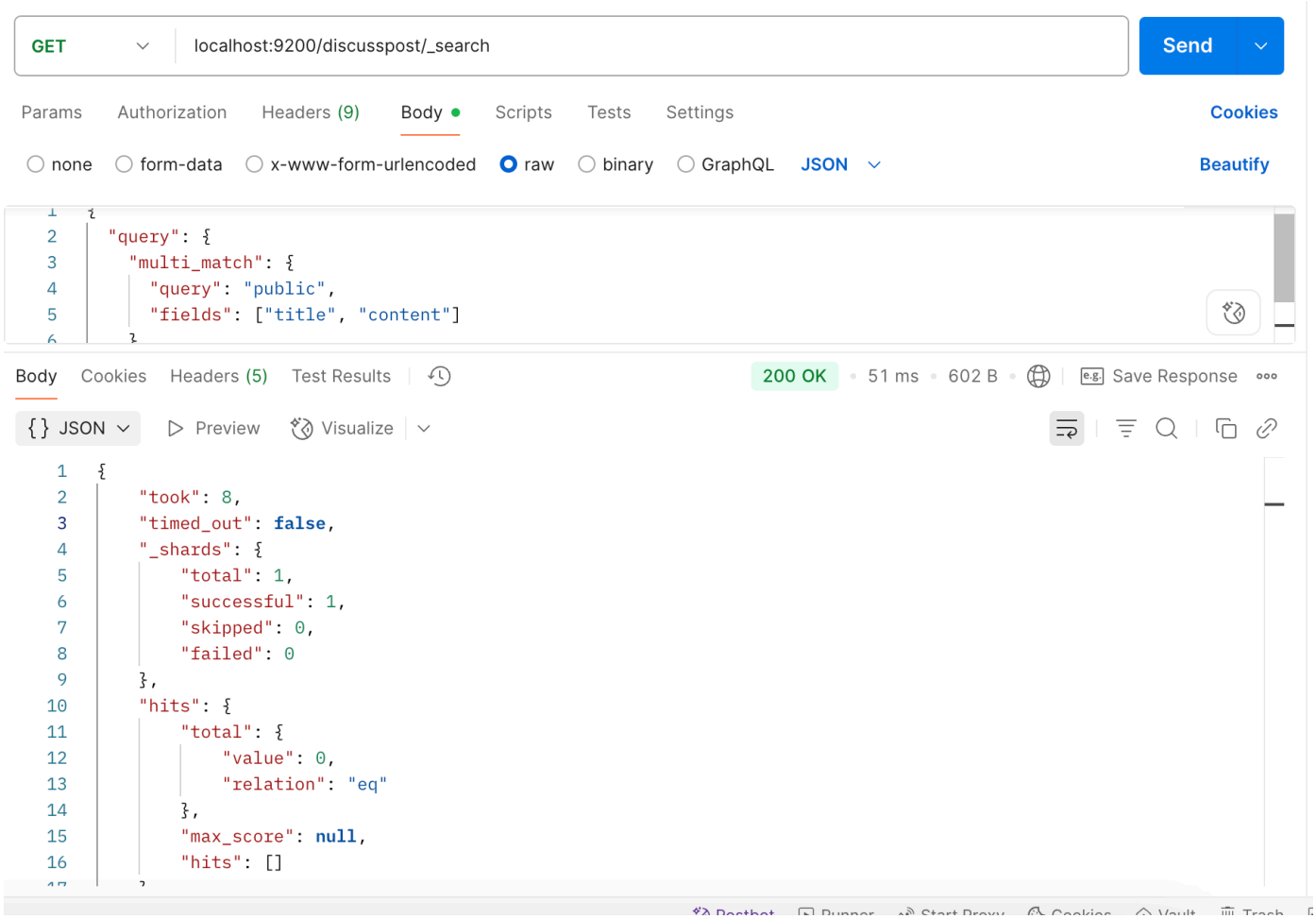
none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify


```
{
  "analyzer": "ik_smart",
  "text": "public"
}
```


Body Cookies Headers (5) Test Results 200 OK 51 ms 567 B Save Response

JSON Preview Visualize


```
{
    "tokens": [
        {
            "token": "public",
            "start_offset": 0,
            "end_offset": 6,
            "type": "ENGLISH",
            "position": 0
        }
    ]
}
```



❓ public搜索，搜索不出来那条帖子

Elasticsearch 对英文和中文混合时，得分机制可能让英文词匹配效果变差，尤其是当其他字段干扰更大时。

Solution: create a new indice

@Document(indexName = "new_discusspost") // ✅ 索引名（类似数据库表名）
public class DiscussPost {

```
@Id  // ✅ 主键
private Long id;

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

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

or

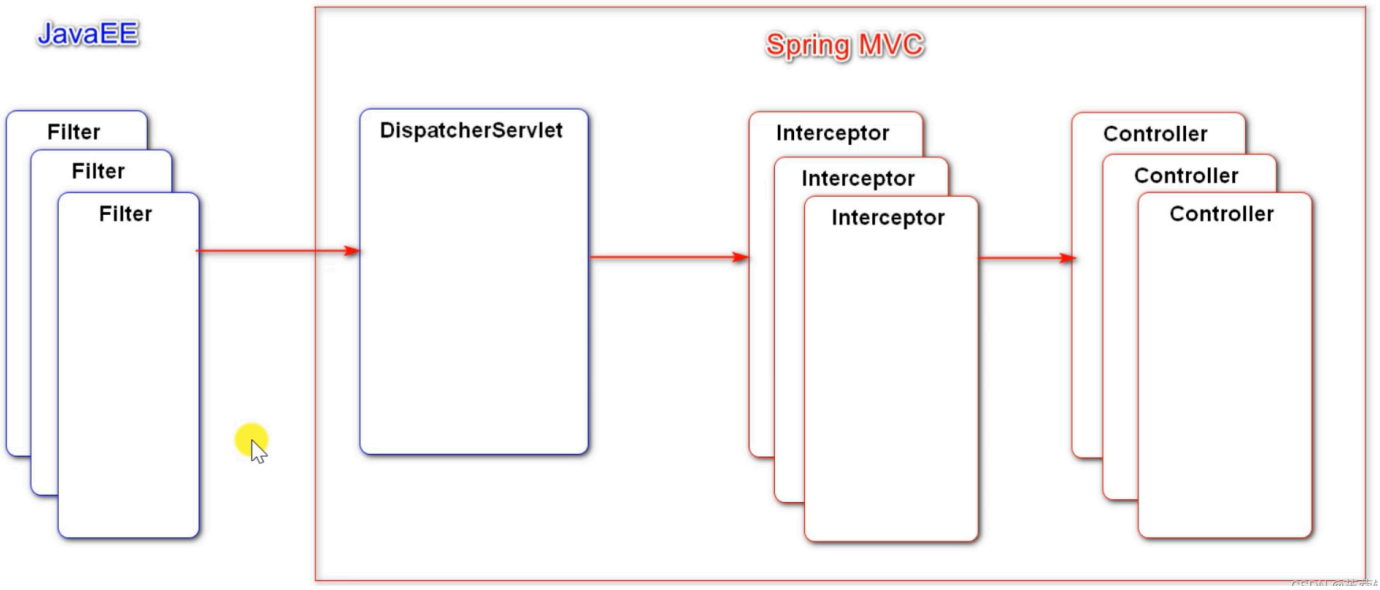
PUT /discusspost_new
{
"mappings": {
"properties": {

```
    "title": {
      "type": "text",
      "analyzer": "ik_max_word",
      "search_analyzer": "ik_smart"
    },
    "content": {
      "type": "text",
      "analyzer": "ik_max_word",
      "search_analyzer": "ik_smart"
    },
    "userId": {
      "type": "integer"
    },
    ...
   }
  }
}
```

analyzer 和 search_analyzer 区别?

名称	用于什么阶段?	举例说明
`analyzer`	存数据时的分词方式	"public post" → [public, post]
`search_analyzer`	搜索查询时的分词方式	搜索 "public" 时，也拆成 [public]

Spring Security - 置顶、加精、删除



```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-security</artifactId>
</dependency>
<dependency>
    <groupId>org.thymeleaf.extras</groupId>
    <artifactId>thymeleaf-extras-springsecurity6</artifactId>
</dependency>
```

Spring Security

Access Control

- **Login Verification**
  - Previously implemented via an interceptor, this basic permission management solution is now deprecated.
  - Use spring secutiry to replace LoginRequiredInterceptor
- **Authorization Configuration**
  - Assign access permissions (e.g., regular user, moderator, administrator) to all requests within the system.
- **Authentication Scheme**
  - Bypass the Security authentication process and adopt the system's original authentication method.
- **CSRF Configuration**
  - Fundamentals of CSRF attack prevention, with configurations for forms and AJAX.

```
<form class="form-inline my-2 my-lg-0" th:action="@{/search}">
    <input class="form-control mr-sm-2" type="search" aria-label="Search" name="keyword"
th:value="${keyword}"/>
    <button class="btn btn-outline-light my-2 my-sm-0" type="submit">Search</button>
</form>
```

// LoginRequiredInterceptor （之前用的 interceptor 是应用层的请求拦截，Spring Security 处理的是安全层的身份权限管理，两个层面不完全一样，但目的类似：控制访问。

```
    String AUTHORITY_USER = "user"; // only logged incan view setting, upload, add,
letter, notice, like, follow/unfollow

    String AUTHORITY_ADMIN = "admin"; // only admin can delete a postset a post as Top

    String AUTHORITY_MODERATOR = "moderator"; // only moderator can set a post as Top
}
```

config

定义哪些请求路径需要认证和授权，哪些可以匿名访问

定义**没有登录**或权限不足时的处理逻辑

配置忽略静态资源的安全拦截

自定义退出登录接口

注入用于持久化 SecurityContext（认证信息）的仓库

UserService（getAuthorities）

LoginTicketInterceptor （用getAuthorities查出该user的权限，是普通用户还是管理员还是，放在hostHoder里）

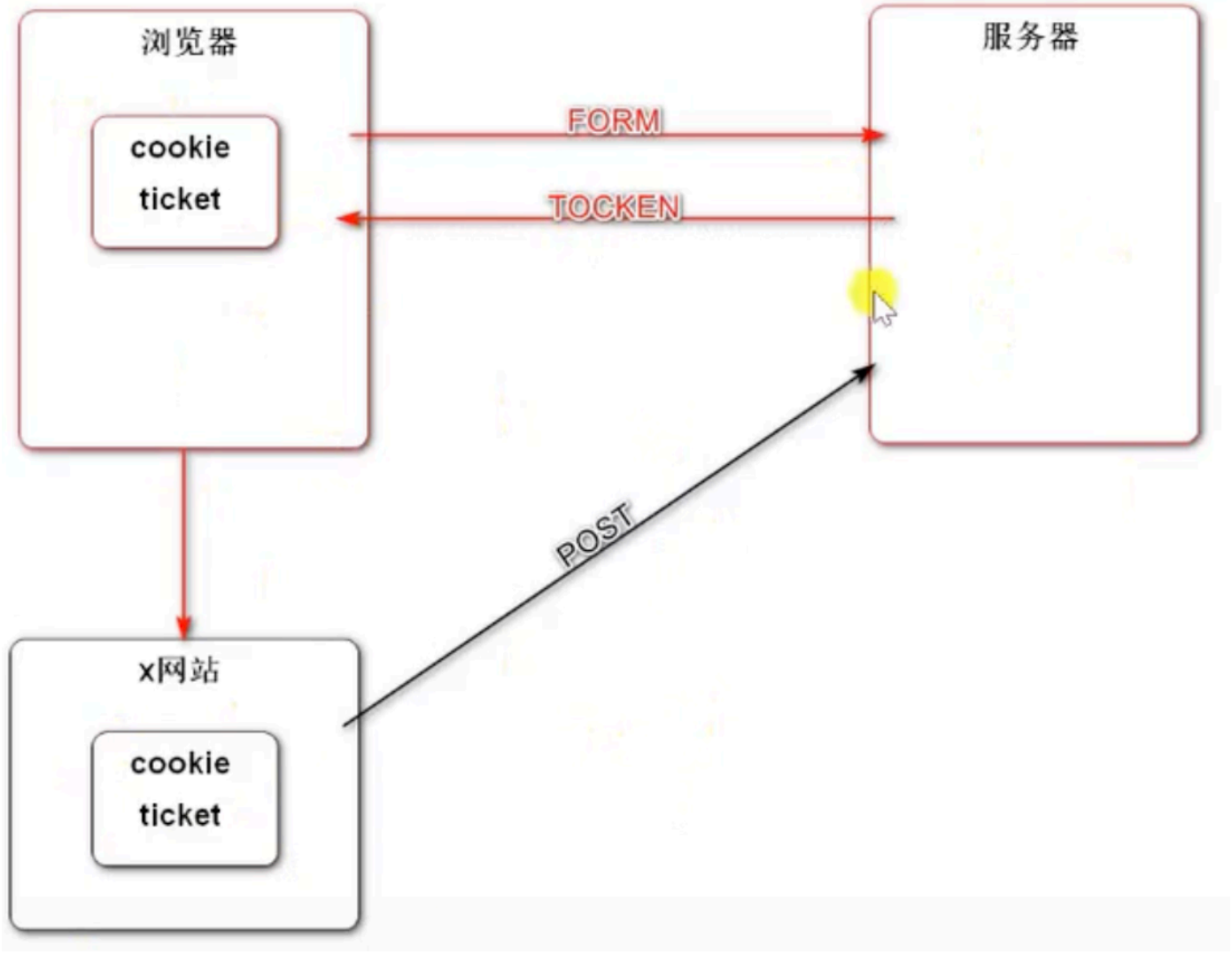
```
    @RequestMapping(path = "/logout", method = RequestMethod.GET)
    public String logout(@CookieValue("ticket") String ticket) {
        userService.logout(ticket);

        // 清理SecurityContextHollder中的权限！！！
        SecurityContextHolder.clearContext();

        return "redirect:/login";
    }
```

LoginTicketInterceptor它本身不决定请求是否放行，只负责“登录状态的初始化”（把数据拿到，然后之后的请求都可以用，取一次数据，一劳永逸的感觉），即“当前是谁”这个上下文的准备。LoginRequiredInterceptor拦截每个进入Controller的请求，没登录就跳到登录页面，没权限就跳错误提示页面（所以被spring security代替了这个功能）

CSRF是一个不好的网站盗取你cookie ticket凭证，假扮你的身份去访问服务器，想服务器提交数据



cookie对，但是token不对

```
<!-- 回帖输入 -->
<div class="container mt-3">
    <form class="replyform" method="post" action="/community/comment/add/234"><input type="hidden" name="_csrf" value="1fef91fe-f2cf-40d5-bf27-570501809a3f" />
```

spring security帮忙生成的，防止rscf攻击的凭证

```
<!--访问该页面时,在此处生成CSRF令牌.-->
<meta name="_csrf" th:content="${_csrf.token}">
<meta name="_csrf_header" th:content="${_csrf.headerName}">
```

index.html

```
// 发送AJAX请求之前,将CSRF令牌设置到请求的消息头中.
//     var token = $("meta[name='_csrf']").attr("content");
//     var header = $("meta[name='_csrf_header']").attr("content");
//     $(document).ajaxSend(function(e, xhr, options){
//         xhr.setRequestHeader(header, token);
//     });
```

Index.js

Top, delete

Nowcoder11，12，13: admin ， 123456，can delete， userType 1

Nowcoder21，22，23，24：，123456

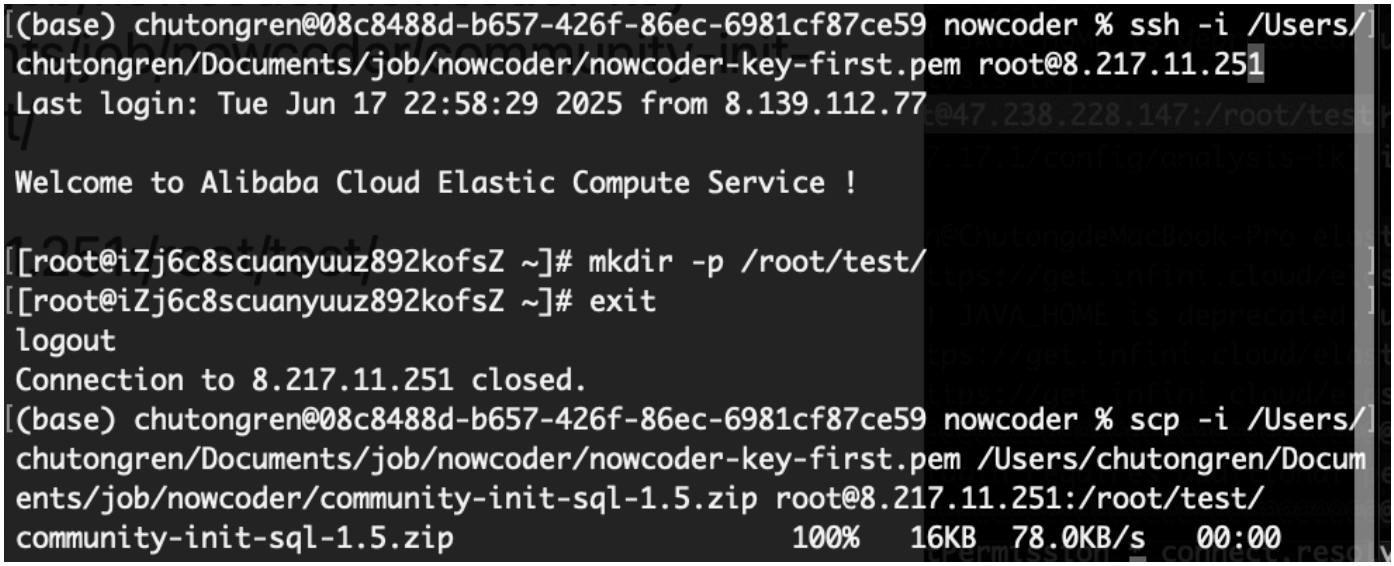
部署项目

Github Education Pack

AWS

Tecent Cloud, Aliyun Cloud free-test 1month

scp -i /Users/chutongren/Documents/job/nowcoder/nowcoder-key-first.pem /Users/chutongren/Documents/job/nowcoder/community-init-sql-1.5.zip root@8.217.11.251:/root/test/



cd ~

yum list unzip*

yum install -y unzip_x86_64

1.

yum list java*

yum install -y -latest-open-jdk

上面的我没用，因为jdk竟然是11版本，太老了吧，用的https://juejin.cn/post/7234428563965247546

tar -zxvf jdk-23.0.2_linux-x64_bin.tar.gz -C /usr/local/jdk

export JAVA_HOME=/usr/local/jdk/jdk-23.0.2

export CLASSPATH=$CLASSPATH:$JAVA_HOME/lib

export PATH=$PATH:$JAVA_HOME/bin

java -version

2.

cd /root

tar -zxvf apache-maven-3.9.10-bin.tar.gz -C /opt

cd /opt/apache-maven-3.9.10/

pwd

配置环境变量etc/profile: vim /etc/profile

CLICK o

export PATH=$PATH:/opt/apache-maven-3.9.10/bin

ESC, :wq

source /etc/profile 使其生效

echo $PATH /usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/root/bin:/opt/apache-maven-3.9.10/bin

3.

yum install -y https://repo.mysql.com/mysql80-community-release-el7-5.noarch.rpm

yum install -y mysql-community-server

rpm --import https://repo.mysql.com/RPM-GPG-KEY-mysql-2023

yum clean all

rm -rf /var/cache/yum

yum makecache

systemctl start mysqld //启动

systemctl status mysql

```
[root@iZj6c8scuanyuuz892kofsZ ~]# systemctl start mysqld
[root@iZj6c8scuanyuuz892kofsZ ~]# systemctl status mysql
Unit mysql.service could not be found.
[root@iZj6c8scuanyuuz892kofsZ ~]# systemctl status mysqld
● mysqld.service - MySQL Server
   Loaded: loaded (/usr/lib/systemd/system/mysqld.service; enabled; vendor preset: disabled)
   Active: active (running) since Wed 2025-06-18 01:31:10 CST; 35s ago
     Docs: man:mysqld(8)
           http://dev.mysql.com/doc/refman/en/using-systemd.html
  Process: 13610 ExecStartPre=/usr/bin/mysqld_pre_systemd (code=exited, status=0/SUCCESS)
 Main PID: 13703 (mysqld)
   Status: "Server is operational"
   CGroup: /system.slice/mysqld.service
           └─13703 /usr/sbin/mysqld

Jun 18 01:31:01 iZj6c8scuanyuuz892kofsZ systemd[1]: Starting MySQL Server...
Jun 18 01:31:10 iZj6c8scuanyuuz892kofsZ systemd[1]: Started MySQL Server.
[root@iZj6c8scuanyuuz892kofsZ ~]#
```

grep 'password' /var/log/mysqld.log
2025-06-17T17:31:04.469074Z 6 [Note] [MY-010454] [Server] A temporary password is generated for root@localhost: OJAokhQYR4*+

alter user root@localhost identified by 'Nowcoder_123';

4

scp -i /Users/chutongren/Documents/job/nowcoder/nowcoder-key-first.pem /Users/chutongren/Documents/job/nowcoder/init_sql.zip root@8.217.11.251:/root/

unzip -d /root init_sql.zip

mysql -u root -p

create database community;

use community;

source /root/init_sql/init_schema.sql;

source /root/init_sql/init_data.sql;

source /root/init_sql/tables_mysql_innodb.sql;

show tables;

SELECT header_url FROM user;

修改用户头像路径，带有localhost存在本机的改成

```
UPDATE user
SET header_url = 'http://images.nowcoder.com/head/666t.png'
WHERE header_url LIKE 'http://localhost%';
```

5

yum list redis*

yum install -y redis.x86_64

systemctl start redis

systemctl status redis

```
[root@iZj6c8scuanyuuz892kofsZ ~]# systemctl start redis
[root@iZj6c8scuanyuuz892kofsZ ~]# systemctl status redis
● redis.service - Redis persistent key-value database
   Loaded: loaded (/usr/lib/systemd/system/redis.service; disabled; vendor preset: disabled)
  Drop-In: /etc/systemd/system/redis.service.d
           └─limit.conf
   Active: active (running) since Wed 2025-06-18 01:56:16 CST; 6s ago
 Main PID: 14508 (redis-server)
   CGroup: /system.slice/redis.service
           └─14508 /usr/bin/redis-server 127.0.0.1:6379

Jun 18 01:56:16 iZj6c8scuanyuuz892kofsZ systemd[1]: Starting Redis persistent key-value database...
Jun 18 01:56:16 iZj6c8scuanyuuz892kofsZ systemd[1]: Started Redis persistent key-value database.
```

redis-cli

keys *

6

Kafka

tar -zxvf kafka_2.13-3.9.0.tgz -C /opt

后台启动kafka

cd /opt/kafka_2.13-3.9.0

bin/zookeeper-server-start.sh -daemon config/zookeeper.properties

nohup bin/kafka-server-start.sh config/server.properties 1>/dev/null 2>&1 &

bin/kafka-topics.sh --list --bootstrap-server localhost:9092

7

tar -zxvf elasticsearch-7.17.1-linux-x86_64.tar.gz -C /opt

unzip -d /opt/elasticsearch-7.17.1/plugins/ik elasticsearch-analysis-ik-7.17.1.zip

cd /opt/elasticsearch-7.17.1/config

Vim elasticsearch.yml

/tmp/elastic/data

/tmp/elastic/logs

vim jvm.options

-Xms256m
-Xmx512m

~esq

啊啊啊啊啊啊啊 安装错了啊啊啊啊啊啊啊 按成arm架构的了 应该是x86

rm -rf /opt/elasticsearch-9.0.2 # 删除错误版本

好吧，不兼容，改成安装https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-7.17.1-linux-x86_64.tar.gz

8

tar -zxvf apache-tomcat-11.0.8.tar.gz -C /opt

vim /etc/profile

Export PATH=$PATH:/opt/apache-tomcat-11.0.8/bin

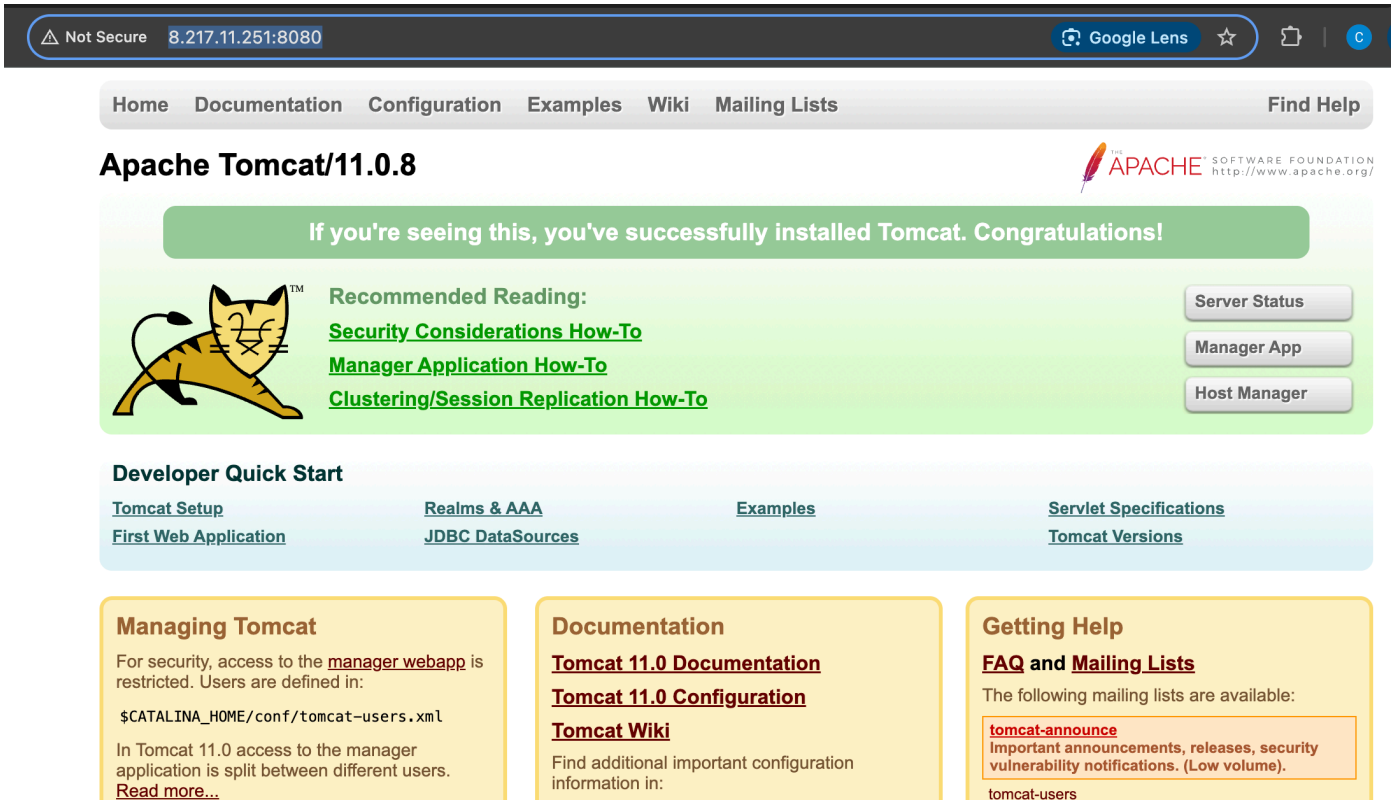
source /etc/profile

startup.sh

```
[root@iZj6c8scuanyuuz892kofsZ bin]# startup.sh
Using CATALINA_BASE:   /opt/apache-tomcat-11.0.8
Using CATALINA_HOME:   /opt/apache-tomcat-11.0.8
Using CATALINA_TMPDIR: /opt/apache-tomcat-11.0.8/temp
Using JRE_HOME:        /usr/local/jdk/jdk-23.0.2
Using CLASSPATH:       /opt/apache-tomcat-11.0.8/bin/bootstrap.jar:/opt/apache-tomcat-11.0.8/bin/tomcat-juli.jar
Using CATALINA_OPTS:
Tomcat started.
```

☐	✓ 允许	1	自定义 TCP	目的: 8080/8080	源: 所有IPv4(0.0.0.0/0)	Tomcat服务	2025年6月17日 22:25:26	编辑 | 复制 | 删除

http://8.217.11.251:8080/



cd /opt/apache-tomcat-11.0.8

Cd webapps

9

yum list nginx*

yum install -y nginx.x86_64

vim /etc/nginx/nginx.conf

upstream myserver {

server 127.0.0.1:8080 max_fails=3 fail_timeout=30s; # 修复拼写：fail_timeout }

server { listen 80; server_name 8.217.11.251; # 修复拼写：server_name location / { proxy_pass http://myserver; proxy_set_header Host $host; proxy_set_header X-Real-IP $remote_addr; } }

systemctl start nginx

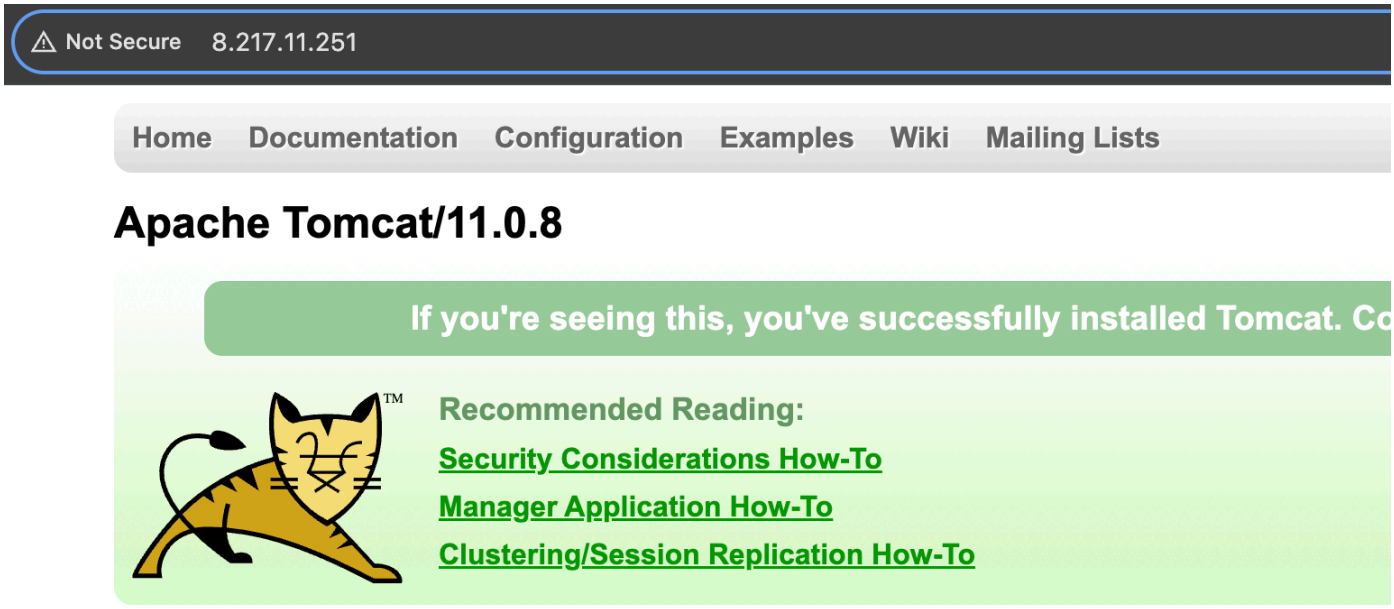
systemctl status nginx

:wq

```
[root@iZj6c8scuanyuuz892kofsZ ~]# systemctl start nginx
[root@iZj6c8scuanyuuz892kofsZ ~]# systemctl status nginx
● nginx.service - The nginx HTTP and reverse proxy server
   Loaded: loaded (/usr/lib/systemd/system/nginx.service; disabled; vendor preset: disabled)
   Active: active (running) since Wed 2025-06-18 04:38:42 CST; 5s ago
  Process: 19985 ExecStart=/usr/sbin/nginx (code=exited, status=0/SUCCESS)
  Process: 19981 ExecStartPre=/usr/sbin/nginx -t (code=exited, status=0/SUCCESS)
  Process: 19979 ExecStartPre=/usr/bin/rm -f /run/nginx.pid (code=exited, status=0/SUCCESS)
 Main PID: 19988 (nginx)
   CGroup: /system.slice/nginx.service
           ├─19988 nginx: master process /usr/sbin/nginx
           ├─19989 nginx: worker process
           └─19990 nginx: worker process

Jun 18 04:38:42 iZj6c8scuanyuuz892kofsZ systemd[1]: Starting The nginx HTTP and reverse proxy server...
Jun 18 04:38:42 iZj6c8scuanyuuz892kofsZ nginx[19981]: nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
Jun 18 04:38:42 iZj6c8scuanyuuz892kofsZ nginx[19981]: nginx: configuration file /etc/nginx/nginx.conf test is successful
Jun 18 04:38:42 iZj6c8scuanyuuz892kofsZ systemd[1]: Started The nginx HTTP and reverse proxy server.
```

☐	✓ 允许	1	自定义 TCP	目的: 80/80	源: 所有IPv4(0.0.0.0/0)	nginx代理	2025年6月17日 22:42:37	编辑 | 复制 | 删除



shutdown.sh

cd /opt/apache-tomcat-11.0.8/

cd webapps/

rm -rf *

改成空的 global.js和pr.

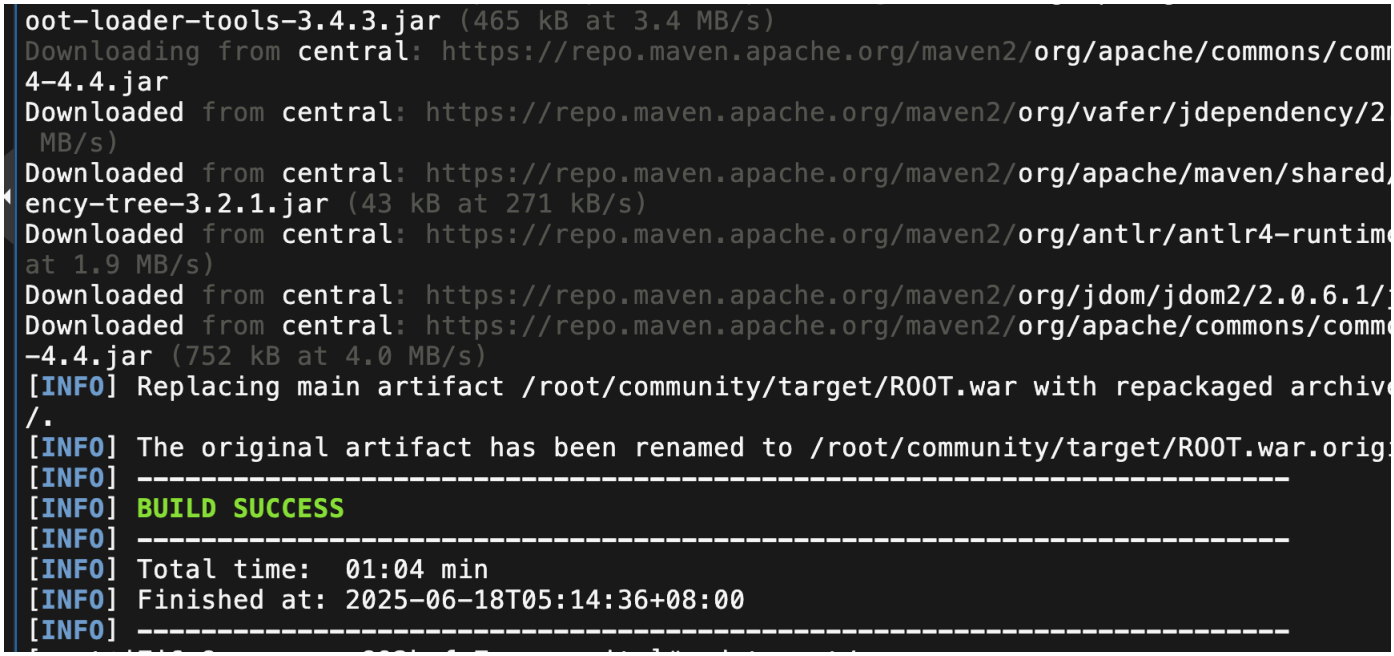
```java
@RequestMapping(path = "/", method = RequestMethod.GET)
public String root() {
    return "forward:/index";
}
```

scp -i /Users/chutongren/Documents/job/nowcoder/nowcoder-key-first.pem /Users/chutongren/Documents/job/nowcoder/community.zip root@8.217.11.251:/root/

unzip -d /root community.zip

cd /community

mvn clean package -Dmaven.test.skip=true

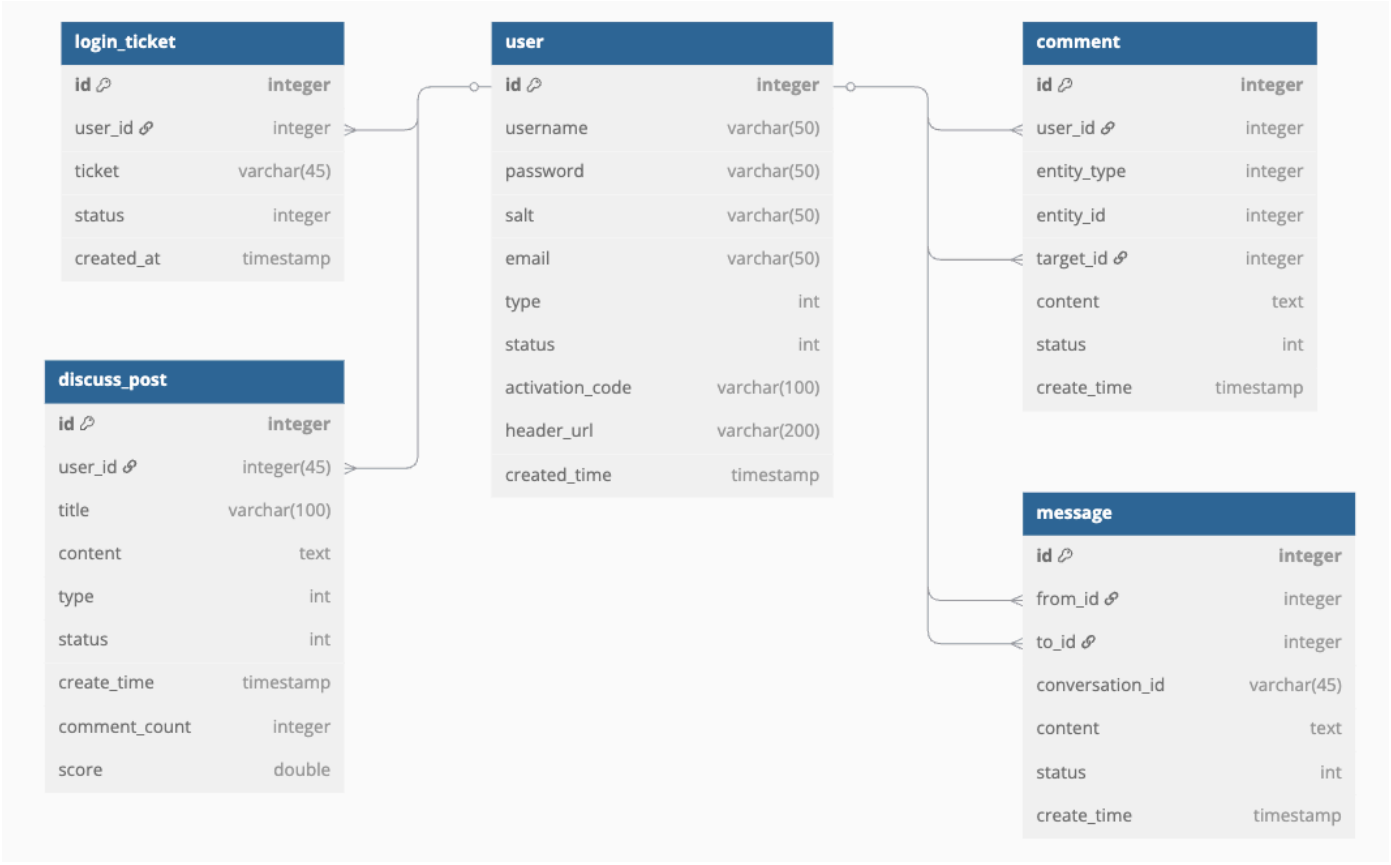


cd target/ ROOT.WAR

mv ROOT.war /opt/apache-tomcat-11.0.8/webapps/

cd /opt/apache-tomcat-11.0.8/webapps/

Tables



// Use DBML to define your database structure

// Docs: https://dbml.dbdiagram.io/docs

```
Table user {
  id integer [primary key]
  username varchar(50)
  password varchar(50)
  salt varchar(50)
  email varchar(50)
  type int
  // 1 = normal user, 2 = admin, 3 = author/poster
  status int
  // 0 = active, 1 = inactive
  activation_code varchar(100)
  header_url varchar(200)
  created_time timestamp
}
```

```
Table login_ticket {
  id integer [primary key]
  user_id integer
  ticket varchar(45) // core
  status integer // 1 = valid
  created_at timestamp
}

Table discuss_post {
  id integer [primary key]
  user_id integer(45)
  title varchar(100)
  content text
  type int
  status int
  create_time timestamp
  comment_count integer
  score double
}

Table comment {
  id integer [primary key]
  user_id integer
  entity_type integer
  entity_id integer
  target_id integer
  content text
  status int
  create_time timestamp
}
```

```
Table message {
  id integer [primary key]
  from_id integer
  to_id integer
  conversation_id varchar(45)
  content text
  status int // 0 = unread, 2 = deleted
  create_time timestamp
}

// user 与 login_ticket 是一对多（一个用户可能对应多个登录票）
Ref: login_ticket.user_id > user.id // many-to-one

// user 与 discuss_post 是一对多（一个用户可以发多个帖子）
Ref: discuss_post.user_id > user.id // many-to-one

// user 与 comment 是一对多（一个用户可以发多个评论）
Ref: comment.user_id > user.id // many-to-one
Ref: comment.target_id > user.id
// comment 与 discuss_post 存在间接引用（comment.entity_type = 帖子，对应 entity_id = discuss_post.id）
// comment.entity_type = 1 → discuss_post.id
// comment.entity_type = 2 → comment.id

// message 表中 from_id 和 to_id 都指向 user
Ref: message.from_id > user.id // many-to-one
Ref: message.to_id > user.id   // many-to-one

// Ref notes.field > users.type
// Ref user_posts: discuss_post.user_id > user.id // many-to-one
// Ref: user.id < follows.following_user_id
```

// Ref: user.id < follows.followed_user_id

✅ 1. `entity_type` / `entity_id` 是不是外键?

是的，它们**组合起来**是实现“多态关联”（polymorphic association）的一种方式。

- `entity_type`：表示评论的对象类型（例如 1 表示帖子的评论，2 表示评论的回复）
- `entity_id`：表示评论的对象 ID，对应 `discuss_post.id` 或 `comment.id`

⚠️ 但它们**不是数据库严格意义上的外键**，因为：

- 数据库外键只能指向一个固定的表（不能动态选择表）
- 所以工具通常不会帮你自动画出连线

✅ 所以：**从逻辑上说是“外键”**，但在数据库设计中，不能直接建外键约束，也不会自动生成连线。

❓ 3. `discuss_post.comment_count` 是不是需要连线?

不需要连线，原因如下：

- `comment_count` 是一个**冗余字段**（denormalized field）
- 它是为了提高查询效率，保存的只是统计值，不是外键或关联字段
- 它的值是通过：

  sql Copy Edit

  ```sql
  SELECT COUNT(*) FROM comment WHERE entity_type = 1 AND entity_id = discuss_post.id
  ```

  计算出来的

- 所以它不需要与 `comment` 表建立外键或连线