CAT 二轮思路

用户

未登录

查询，查看

首页展示五个问题，点赞量排序，分页

问题显示标题，提问者，和点赞量

查看问题，答案，评论。操作时弹出登录界面

可查询，模糊查询，分页

登录

登录注册操作

1. 注册与登录:

- - 用户只通过邮箱方式注册账号，并向用户发送验证邮件包含验证码等信息。
  - 登录注册使用图形验证码【点击可更换验证码】
  - 忘记密码可以用邮箱发送验证码来修改密码
- 注册后有默认用户名和默认头像
- 用户可以修改个人信息
- 登录时可以使用
- 1. 邮箱+密码
  2. 用户名+密码
- 忘记密码使用邮箱验证码修改密码

权限

权限设置：

首先，利用dao，把一个用户的所有权限都查询出来。再把查询出来的权限列表，构建成权限表，返回表

这样，用户登录后，获取权限的核心逻辑就算完成了。前端解析权限，构建页面后，用户就只能看到他有权限的东西，他没有权限的菜单或是功能，在页面上就不会展示出来。那么，为了考虑服务端的性能，用户登录一次获取了权限之后，就会把权限保存到redis，下次，用户再登录的时候，就不需要再从服务端获取权限了。但是，用户的权限，是随时会发生改变的。所以客户端的权限，要有更新的功能，问题是什么时候更新？这个时候在改变权限的时候删除redis的权限字段。

关注

支持关注和取消关注其他**用户**、**话题**等

- - 可以搜索用户，并且查看用户信息，用户发布的问题贴等。
  - 关注后，可以查看与对方的共同关注列表。

动态界面

里面是关注的人发布的动态。

滚动分页

私信

里面是别人发送的私信

可以设置不接受私信，或接受已关注

收藏夹

里面是收藏的文章。

聊天室

谁创建谁群主，只有群主可以拉人。只能拉关注了群主的人

其他人可以退群

群主可以解散群

群主可以禁言群成员，群主支持添加、删除管理员。

自己发布问题

- 用户可以提出问题，并设置问题标题、问题内容、分类等信息，并支持图片上传等附加功能。

有人回答就有私信

浏览记录

记录最近五天的100条

拉黑

- 被答主拉黑的用户不能向答主发送**私信**，也不能在回答下面**评论。**

举报功能

- 可对up主的问题、问题的回答、以及回答下面的评论进行举报，需要输入举报理由。被举报的内容将会被管理员核实后不会被推送。

管理员

举报核验

- - 通过获取用户的举报信息，可以**对用户进行封禁**，被封禁的用户之后只能查看问题,、回答和评论【相当于变成下边的游客身份】
  - 发布的问题需要进行审核，只有审核通过的问题才能被用户查看到（如果审核失败需要向发布者告知失败原因）
  - 管理员可以对用户进行权限管理，如发布问题、回答、点赞等权限的控制。根据用户不同的反馈进行不同的封禁，并且设定封禁截止日期，比如禁止用户评论，或者禁止用户发布问题，也有可能该用户同时不能评论或者发布问题，并且封禁的时效不一致（评论一天后解封，发布问题两天后）
  - 敏感词汇校验
    - 发布的问题、回答、以及评论，后台检测是否存在**敏感词汇**，如诋毁党、色情等信息。若存在，则系统将自动驳回发布请求并提醒发布者

日记

4.11

扒了点前端的代码

4.12

创建了数据库

4.13

又是没有见过的bug

```
Exception in thread "main" java.lang.NoClassDefFoundError Create breakpoint : javax/servlet/http/HttpServlet
    at java.lang.ClassLoader.defineClass1(Native Method)
    at java.lang.ClassLoader.defineClass(ClassLoader.java:756)
    at java.security.SecureClassLoader.defineClass(SecureClassLoader.java:142)
    at java.net.URLClassLoader.defineClass(URLClassLoader.java:473)
    at java.net.URLClassLoader.access$100(URLClassLoader.java:74)
    at java.net.URLClassLoader$1.run(URLClassLoader.java:369)
    at java.net.URLClassLoader$1.run(URLClassLoader.java:363) <1 个内部行>
    at java.net.URLClassLoader.findClass(URLClassLoader.java:362)
    at java.lang.ClassLoader.loadClass(ClassLoader.java:418)
    at sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:355)
    at java.lang.ClassLoader.loadClass(ClassLoader.java:351)
    at java.lang.Class.forName0(Native Method)
    at java.lang.Class.forName(Class.java:348)
    at com.my_framework.www.utils.ClassUtil.loadClass(ClassUtil.java:38)
    at com.my_framework.www.utils.ClassUtil.doAddClass(ClassUtil.java:139)
    at com.my_framework.www.utils.ClassUtil.addClass(ClassUtil.java:115)
    at com.my_framework.www.utils.ClassUtil.addClass(ClassUtil.java:127)
    at com.my_framework.www.utils.ClassUtil.getClassSet(ClassUtil.java:63)
    at com.my_framework.www.beans.BeanDefinitionReader.<init>(BeanDefinitionReader.java:44)
    at com.my_framework.www.context.Impl.ApplicationContextImpl.refresh(ApplicationContextImpl.java:48)
    at com.my_framework.www.context.Impl.ApplicationContextImpl.<init>(ApplicationContextImpl.java:40)
```

排查了一下，发现把

```
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>javax.servlet-api</artifactId>
    <version>4.0.1</version>
</dependency>
```

的删了就可以

正常运行，但是一直学的就是加上protected啊。

运行web项目时，要先包扫描，才可以注入，所以我们要在初始化tomcat的时候加上包扫描

```
resh helloServiceImpl
resh com.huangTaiQi.www.service.HelloService
resh userControllerImpl
resh com.huangTaiQi.www.controller.IUserController
resh noArgsServiceImpl
resh com.huangTaiQi.www.service.NoArgsService
resh hiServiceImpl
resh com.huangTaiQi.www.service.HiService
ulateBean class com.huangTaiQi.www.service.impl.HelloServiceImpl  autowired:com.huangTaiQi.www.service.impl.HiServiceImpl
ulateBean 1111111null
ulateBean class com.huangTaiQi.www.service.impl.HelloServiceImpl  autowired:com.huangTaiQi.www.service.impl.HiServiceImpl
ulateBean class com.huangTaiQi.www.controller.impl.UserControllerImpl  autowired:com.huangTaiQi.www.service.impl.HelloServiceImpl
ulateBean class com.huangTaiQi.www.controller.impl.UserControllerImpl  autowired:com.huangTaiQi.www.service.impl.HelloServiceImpl
r occurred in com.huangTaiQi.www.controller.impl.UserControllerImpl e = {0}
```

```
    java.lang.reflect.InvocationTargetException <4 个内部行>
        at com.huangTaiQi.www.controller.BaseController.service(BaseController.java:39)
        at javax.servlet.http.HttpServlet.service(HttpServlet.java:582) <22 个内部行>
    Caused by: java.lang.NullPointerException
        at com.huangTaiQi.www.controller.impl.UserControllerImpl.hello(UserControllerImpl.java:42)
        ... 28 more
已初始化
hello
```

为什么我已经将helloServiceImpl注入进UserController，但是还是有空指针异常?

```
field: "com.huangTaiQi.www.service.impl.HelloServiceImpl com.huangTaiQi.www.controller.impl.UserControllerImpl.helloService"
```

我明白了

```
System.out.println(this);
System.out.println(this.equals(applicationContext.getBean( beanName: "userControllerImpl")));
```

```
com.huangTaiQi.www.controller.impl.UserControllerImpl@4b4cbd35
false
```

不是同一个类。

我试一下能不能用在userController用单例解决

消息 实例化Servlet类[com.huangTaiQi.www.controller.impl.UserControllerImpl]异常

描述 服务器遇到一个意外的情况，阻止它完成请求。

例外情况

```
javax.servlet.ServletException: 实例化Servlet类[com.huangTaiQi.www.controller.impl.UserControllerImpl]异常
        org.apache.catalina.authenticator.AuthenticatorBase.invoke(AuthenticatorBase.java:494)
        org.apache.catalina.valves.ErrorReportValve.invoke(ErrorReportValve.java:93)
        org.apache.catalina.valves.AbstractAccessLogValve.invoke(AbstractAccessLogValve.java:682)
        org.apache.catalina.connector.CoyoteAdapter.service(CoyoteAdapter.java:367)
        org.apache.coyote.http11.Http11Processor.service(Http11Processor.java:639)
        org.apache.coyote.AbstractProcessorLight.process(AbstractProcessorLight.java:63)
        org.apache.coyote.AbstractProtocol$ConnectionHandler.process(AbstractProtocol.java:932)
        org.apache.tomcat.util.net.NioEndpoint$SocketProcessor.doRun(NioEndpoint.java:1695)
        org.apache.tomcat.util.net.SocketProcessorBase.run(SocketProcessorBase.java:49)
        org.apache.tomcat.util.threads.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1191)
        org.apache.tomcat.util.threads.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:659)
        org.apache.tomcat.util.threads.TaskThread$WrappingRunnable.run(TaskThread.java:61)
        java.lang.Thread.run(Thread.java:750)
```

根本原因。

```
java.lang.NoSuchMethodException: com.huangTaiQi.www.controller.impl.UserControllerImpl.<init>()
        java.lang.Class.getConstructor0(Class.java:3082)
        java.lang.Class.getConstructor(Class.java:1825)
        org.apache.catalina.authenticator.AuthenticatorBase.invoke(AuthenticatorBase.java:494)
        org.apache.catalina.valves.ErrorReportValve.invoke(ErrorReportValve.java:93)
        org.apache.catalina.valves.AbstractAccessLogValve.invoke(AbstractAccessLogValve.java:682)
        org.apache.catalina.connector.CoyoteAdapter.service(CoyoteAdapter.java:367)
        org.apache.coyote.http11.Http11Processor.service(Http11Processor.java:639)
        org.apache.coyote.AbstractProcessorLight.process(AbstractProcessorLight.java:63)
        org.apache.coyote.AbstractProtocol$ConnectionHandler.process(AbstractProtocol.java:932)
        org.apache.tomcat.util.net.NioEndpoint$SocketProcessor.doRun(NioEndpoint.java:1695)
        org.apache.tomcat.util.net.SocketProcessorBase.run(SocketProcessorBase.java:49)
        org.apache.tomcat.util.threads.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1191)
        org.apache.tomcat.util.threads.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:659)
        org.apache.tomcat.util.threads.TaskThread$WrappingRunnable.run(TaskThread.java:61)
        java.lang.Thread.run(Thread.java:750)
```

我笑了。我懂了，在转发器改

4.14

完善beanUtils

4.15

如果mysql中储存了无符号的int，就会导致java中的Integer溢出，那么unsigned int被转成Long就非常合理且安全了。

4.18

电脑故障了，自动重启了，导致我前几天的**笔记丢失**了。学到了学完就保存

补一下重点的笔记

1. 解决**threadlocal拿不到值**的问题

   因为在web项目中一个请求进来系统会在线程池中选择线程执行，所以之前在threadlocal中拿的值可能就会拿不到。

   解决办法：在filter中将值存入threadlocal，当前请求就可以轻松地拿到了

   注意：这就导致了一个问题，ThreadLocal在没有外部强引用时，发生GC时会被回收，如果创建ThreadLocal的线程一直持续运行，那么这个Entry对象中的value就有可能一直得不到回收，发生内存泄露。

   就比如线程池里面的线程，线程都是复用的，那么之前的线程实例处理完之后，出于复用的目的线程依然存活，所以，ThreadLocal设定的value值被持有，导致内存泄露。

   记得在使用的最后用remove把值清空

实现了@Transaction注解，使用cglib动态代理实现事务管理。

有一点可以优化的就是没有管方法，全部是数据库和redis都开启事务，有时候会浪费资源，

报了下面的错

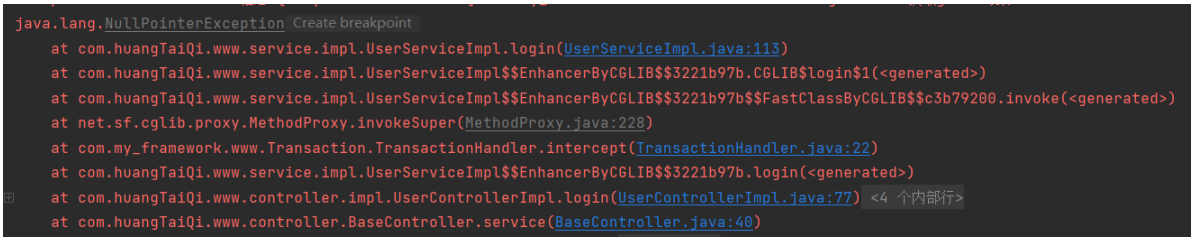
```
redis.clients.jedis.exceptions.JedisException: Could not get a resource from the pool
```

这是一个很神奇的错误，因为我很清楚我的ip和密码是正确的。

所以我改了什么会使它报错呢？看看日志好像是连接池太小了。

使用cglib动态代理实现事务管理后和jdk代理发生了相似的错误。代理对象确实不可以拿到被代理的对象的元素。

那么如何解决这个问题呢？



cglib动态代理实现事务管理，代理类拿不到被代理类被注入的对象，因为被注入的成员变量在被代理时没有初始化，ioc是在获取到实例后再注入的，然后cglib会调用构造方法实现代理，所以cglib动态代理使用被注入的成员变量时报空指针，怎么解决？

4.19

乱搞git代码没了。。。。

折磨

学到了不要乱搞git，

每一个小功能都要上传

4.20

写请求转发

4.21

写请求转发

4.22

固定虚拟机ip，改JDK代理的bug

4.23

改请求转发的bug

改事务bug

如果在 Redis 的事务中使用 Jedis 命令，当调用 `multi()` 方法时，Jedis 将进入 Multi 状态。在 Multi 状态下，不能使用 Jedis 的其他方法。因此，如果尝试在这种情况下使用 Jedis 的其他方法，如 `set()`、`get()` 等，Jedis 将返回错误消息：“Cannot use Jedis when in Multi. Please use Transaction or reset jedis state.”

4.24

webSocket

WebSocket 请求和常规的 HTTP 请求不同，不需要通过请求转发器。WebSocket 是一种基于 TCP 的协议，允许客户端和服务器之间进行双向通信。当客户端发送 WebSocket 请求时，它将直接连接到服务器，并打开一个持久化的连接，从而允许服务器向客户端推送消息，而无需客户端请求。因此，WebSocket 请求不会经过请求转发器。在使用WebSocket时，一般不需要在 web.xml 文件中进行特殊的配置。

4.25

前端限制我的发挥

4.26

在使用WebSocket实现群聊时，对于某些没有上线的用户，使用数据库来缓存他们的消息。当用户上线时，从数据库中获取这些消息并将其发送给用户。

当用户登录时，获取该用户的所在的所有群聊，推送群聊信息到redis中，前端通过当前所在页面的群聊拉取消息并显示。

步骤：

1. 创建一个消息队列：使用redis为每个用户创建一个队列，或者为每个群聊创建一个队列。
2. 当用户发送消息时，将消息添加到消息队列中。
3. 当用户上线时，从消息队列中获取消息。
4. 将消息发送给用户。

历史记录用jedis存七天

4.27

写群聊

4.28

写群聊

4.29

```
   private Map<Long, List<CommentEntity>> groupCommentsByTid(List<CommentEntity>
commentList) {
//        Map<Long, List<CommentEntity>> commentsMap = new HashMap<>();
//
//        // 将评论按照tid分组，存储在commentsMap中
//        for (CommentEntity comment : commentList) {
//            // 如果tid在commentsMap中不存在，则新建一个ArrayList作为value
//            commentsMap.computeIfAbsent(comment.getTopId(), k -> new
ArrayList<>()).add(comment);
//        }
//
//        return commentsMap;
        return
commentList.stream().collect(Collectors.groupingBy(CommentEntity::getTopId));
    }
```

明显看出stream的漂亮

4.30

写业务

5.1

写业务

5.2

写业务

5.3

摸鱼

5.4

回学校