**Echo — 开源社区系统**

**专业：软件工程11809**

**学号：1804230917**

**姓名：马宜南**

**武汉纺织大学**

**计算机与人工智能学院**

**2021 年 6 月**

目 录

[1 引言 4](#_Toc77312655)

[2 系统技术概括 4](#_Toc77312656)

[2.1 系统前端技术概括 4](#_Toc77312657)

[2.1.1 Bootstrap 框架 4](#_Toc77312658)

[2.1.2 Jquery 及 Ajax 4](#_Toc77312659)

[2.2 系统后端技术概括 7](#_Toc77312660)

[2.2.1 MySQL 数据库 7](#_Toc77312661)

[2.2.2 Tomcat 服务器 8](#_Toc77312662)

[2.2.3 MVC 设计模式 9](#_Toc77312663)

[2.2.4 Spring 框架 10](#_Toc77312664)

[2.2.5 SpringMVC 框架 10](#_Toc77312665)

[2.2.6 MyBatis 框架 11](#_Toc77312666)

[3 系统设计 12](#_Toc77312667)

[3.1 数据库 12](#_Toc77312668)

[3.2 账号 15](#_Toc77312669)

[3.3 帖子 17](#_Toc77312670)

[3.4 搜索 20](#_Toc77312671)

[3.5 数据统计 21](#_Toc77312672)

[4 总结与展望 22](#_Toc77312673)

[参考文献 23](#_Toc77312674)

# 引言

出于个人实际的学习经历，学习过程中刚开始难免会有无从下手的困扰，书本的内容滞后于当下的学习，网络博客的内容提供者与接受者学习时间不够同步。

在GitHub上找到一个开源的开源社区系统的项目，让想学习的小伙伴在一起讨论，分享者即有反馈，学习者即有答疑。

# 系统技术概括

## 系统前端技术概括

### Bootstrap 框架

Bootstrap，来自 Twitter，是目前最受欢迎的前端框架。Bootstrap 是基于 HTML、CSS、JAVASCRIPT 的，它简洁灵活，使得 Web 开发更加快捷。

Bootstrap 框架包含内容：

* **基本结构**：Bootstrap 提供了一个带有网格系统、链接样式、背景的基本结构。这将在Bootstrap 基本结构部分详细讲解。
* **CSS**：Bootstrap 自带以下特性：全局的 CSS设置、定义基本的 HTML 元素样式、可扩展的 class，以及一个先进的网格系统。这将在Bootstrap CSS部分详细讲解。
* **组件**：Bootstrap 包含了十几个可重用的组件，用于创建图像、下拉菜单、导航、警告框、弹出框等等。这将在布局组件部分详细讲解。
* **JavaScript 插件**：Bootstrap包含了十几个自定义的jQuery 插件。您可以直接包含所有的插件，也可以逐个包含这些插件。这将在Bootstrap插件部分详细讲解。
* **定制**：您可以定制Bootstrap的组件、LESS 变量和jQuery 插件来得到您自己的版本。

### Jquery 及 Ajax

Jquery 是一个快速、简洁的 JavaScript 框架，其倡导写更少的代码，做更多的事情。它封装 JavaScript 常用的功能代码，提供一种简便的 JavaScript 设计模式，优化 HTML 文档操作、事件处理、动画设计和 Ajax 交互[4]。

Jquery 的核心特性可以总结为：具有独特的链式语法和短小清晰的多功能接口；具有高效灵活的 CSS 选择器，并且可对 CSS 选择器进行扩展；拥有便捷的插件扩展机制和丰富的插件。

一般而言，选择使用 Jquery 是因为它的以下特点：

(1)更方便的选择器

Jquery 的选择机制构建于 CSS 的选择器，它提供了快速查询 DOM 文档中元素的能力，而且大大强化了 JavaScript 中获取页面元素的方式。相比于原本的选择器，Jquery 使用更少的语句即能达到相同的效果，同时选择的精度也要更高。

(2)提供页面动态效果

Jquery 中内置了一系列的动画效果，比如淡入淡出、元素移除等动态特效。这些特效增强了用户在浏览网页时的体验感，许多的网站都有使用。

(3)创建 Ajax 无刷新网页

Ajax 是异步的 JavaScript 和 XML 的简称，可以协助开发出非常灵敏无刷新的网页

(4)对 JavaScript 语言的增强

Jquery提供了对基本 JavaScript 结构的增强，比如元素迭代和数组处理等操作。

(5)增强的事件处理

Jquery 提供了各种页面事件，它可以避免程序员在 HTML 中添加太多事件处理代码，同时它的事件处理器消除了各种浏览器兼容性问题。

Ajax 是 Jquery 框架的一部分功能简称，它在网页制作中应用广泛，其具体

功能在上面叙述 Jquery 的特点中也有提到，即可以使我们发送请求时进行异步操作，在不刷新页面的情况下对后台的处理产生响应。

不借助 Ajax 的话，我们在页面逻辑操作时，必须要进行从当前页面到控制视图层再返回当前页面的多次跳转，很影响用户的操作体验。而借助 Ajax，后端在处理请求时用户依然可以在前端页面继续进行浏览，直到响应返回前端用户再进行操作，整个过程更加连贯，用户的操作体验自然也会更好。

## 系统后端技术概括

### MySQL 数据库

数据库分为关系型数据库和非关系型数据库。

关系型数据库指的是采用了关系模型来组织数据的数据库，拥有关系，元组， 属性，域，关键字，关系模式等概念。关系型数据库比较直观，更容易被理解， 使用起来较为方便，易于维护。但关系型数据库性能欠佳，在处理大量关系复杂的数据时，效率很低。我们常用的关系型数据库有 [Oracle](https://www.oracle.com/database/index.html)，[Microsoft SQL Server](https://www.microsoft.com/en-us/sql-server/)， MySQL，[PostgreSQL](https://www.postgresql.org/)，[DB2](https://www.ibm.com/analytics/us/en/db2/)，[Microsoft Access](https://products.office.com/zh-cn/access)，[SQLite](https://www.sqlite.org/) 等[5]。

非关系数据库指的是指与关系型数据库相对的非关系型的，分布式的数据库。它以键值对存储，且结构不固定，每一个元组可以有不一样的字段，每个元组可 以根据需要增加一些自己的键值对，不局限于固定的结构。非关系型的数据库结 构简单，处理数据时可以节省一点的时间和开销。但它仅限处理一些简单的没有 复杂关联关系的数据库，不适合持久存储海量数据。由于非关系型数据库都是针 对某些特定的应用需求出现的，因此，不同非关系数据库的特长也有所不同，主 要分为以下几类：面向高性能并发读写的 key-value 数据库，主流代表为 [Redis](https://redis.io/)， [Amazon DynamoDB](https://aws.amazon.com/cn/dynamodb/) ； 面向海量数据访问的面向文档数据库， 主流代表为[MongoDB](https://www.mongodb.com/)；面向搜索数据内容的搜索引擎，主流代表为 [Splunk](https://www.splunk.com/zh-hans_cn)，[Solr](https://lucene.apache.org/solr/)；面向可扩展性的分布式数据库，主流代表为 [Cassandra](https://cassandra.apache.org/)，[HBase](https://hbase.apache.org/)[6]。

MySQL 是典型的关系型数据库，它由瑞典 MySQL AB 公司开发，在被收购之后属于 [Oracle](https://baike.baidu.com/item/Oracle) 旗下产品。与其他关系型数据库相同，MySQL 将数据保存在不同的表中，确保效率并提高了灵活性。MySQL 使用 C 和 [C++](https://baike.baidu.com/item/C%2B%2B)编写，并使用了多种[编译器](https://baike.baidu.com/item/%E7%BC%96%E8%AF%91%E5%99%A8)进行测试，保证了[源代码](https://baike.baidu.com/item/%E6%BA%90%E4%BB%A3%E7%A0%81)的可移植性，使得它能够支持大多数的操作系统。

MySQL 所使用的操作语言 是用于访问[数据库](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%BA%93/103728)的最常用标准化语言 SQL 语言。由于其体积小、速度快、总体拥有成本低，尤其是[开放源码](https://baike.baidu.com/item/%E5%BC%80%E6%94%BE%E6%BA%90%E7%A0%81/7176422)这一特点，一般中小型网站的开发都选择 MySQL 作为网站数据库。虽然 MySQL 与其他的大型数据库比有所不足，但对于个人开发者，MySQL 数据库包含了个人开发所需的全部功能，并且它是免费开放的，降低了学习成本。也正是出于上述原因，我在房屋租赁系统中选择了 MySQL 数据库为项目的数据库。

### Tomcat 服务器

搭建 Web 项目，服务器是必不可少的。而 Tomcat 技术先进、性能稳定， 而且免费，是个人 JavaWeb 开发的首选。

Tomcat 服务器是一个免费的开放源代码的 Web 应用服务器，由 [Apache](https://baike.baidu.com/item/Apache/6265)、Sun 和其他一些公司及个人共同开发而成，属于轻量级应用[服务器](https://baike.baidu.com/item/%E6%9C%8D%E5%8A%A1%E5%99%A8)。它更多在中小型系统和并发访问用户不是很多的场合下被普遍使用，是开发和调试 JSP 程序的首选。在 Tomcat 配置完成后，开发者就可以在服务器上运行 JSP 页面和 Servlet。它具有处理 HTML 页面请求的功能，同时还是一个 Servlet 和 JSP 容器。

我在项目中搭建的是 Tomcat 7.0 的版本，个人使用时经过一些简单的配置配合 Idea、Eclipse 等 Java 开发软件就能较为方便地使用。

### MVC 设计模式

MVC 是软件工程中的一种架构模式，它的全称是 Model View Controller， 其实即是 Model（模型），View（视图），Controller（控制器）的组合。它将业务逻辑聚集到一个部件里面，在改进和个性化定制界面及用户交互的同时， 不需要重新编写业务逻辑。

在 Web 应用程序中，MVC 设计模式的三个部分有着各自的业务逻辑：

Model（模型层）：Model（模型）是应用程序中用于处理应用程序数据逻辑的单位对象，通常模型对象负责在数据库中存取数据。举例来说，数据库中每一个数据表都可以对应一个模型，该模型与数据表拥有相同的属性。在我们需要在该数据表中获取数据时，该模型就会被使用。

View（视图层）：View（视图）是应用程序处理数据后进行显示的部分，它提供一个可视化的部分，来较为直观的在用户数据处理完成后产生响应。

Controlle（r 控制层）：Controller(控制器)是应用程序中处理用户交互的部分。

通常控制器负责从视图读取数据，控制用户输入，并向模型发送数据。

MVC 分层有助于管理复杂的应用程序，因为您可以在一个时间内专门关注一个方面。例如，您可以在不依赖业务逻辑的情况下专注于视图设计。同时也让应用程序的测试更加容易。

MVC 分层同时也简化了分组开发。不同的开发人员可同时开发视图、控制器逻辑和业务逻辑。

我在项目中使用的 SSM 框架也是基于 MVC 设计模式的。SSM 是 Spring、SpringMVC、MyBatis 的首字母缩写，因为 SpringMVC 是 Spring 中的部分内容， 所以 SSM 框架可以说由 Spring、MyBatis 两个开源框架整合而成。SSM 框架经常作为数据源较简单的 Web 项目的框架，在下面我也将对 SSM 框架进行一个大概的描述。

### Spring 框架

Spring 框架用于简化软件开发而创建，是 JavaWeb 开发的常用框架。它的主要功能是使用 JavaBean 代替 EJB（Enterprise Java Beans 的技术简称，是之前的应用开发组件）。

在 Spring 框架之前，开发过程中编写的代码经常出现类似或者重复的情况，这导致开发过程复杂，开发效率和性能都不太理想。Spring 的出现就是为了解决早期这种应用开发的情况，它致力于提供一个以统一的、高效的方式构造整个应用，并且可以将单层框架以最佳的组合揉和在一起建立一个连贯的体系。

Spring 作为一个容器框架，其最大的两个特性是控制反转（IOC）和面向切面（AOP）。

控制反转：控制反转是 Spring 框架作为容器框架的主要特性。原本我们在Java 开发中，我们要使用对象时都需要自身创建对象再进行操作。而使用Spring 框架后，所需对象依赖的其它对象会通过被动的方式传递进来，而不是这个对象自己创建或者查找依赖对象，方便开发者使用对象的同时节省空间， 促进了松耦合。

面向切面：面向切面是[函数式编程](https://baike.baidu.com/item/%E5%87%BD%E6%95%B0%E5%BC%8F%E7%BC%96%E7%A8%8B/4035031)的一种衍生范型。利用面向切面可以对业务逻辑的各个部分进行隔离，从而使得业务逻辑各部分之间的[耦合度](https://baike.baidu.com/item/%E8%80%A6%E5%90%88%E5%BA%A6/2603938)降低， 提高程序的可重用性，同时提高了开发的效率。

在项目中使用 Spring 框架需要在项目中配置部分的配置文件，一般在项目中会配置文件搭配注解使用，简化配置并方便进行管理。

### SpringMVC 框架

SpringMVC 是 Spring 框架的后续产品，它在 SSM 框架中的主要作用作为

Controller（控制器）部分处理 View（视图层）。SpringMVC 分离了控制器、模型对象、过滤器以及处理程序对象的角色，这种分离让它们更容易进行定制。

在实际的使用中，它的作用类似于 Servlet。配置文件配置完成后，创建对应的文件，并一一添加注解，通过这些注解的引用，前端的请求就能转到对应的方法进行处理。但不同的是，SpringMVC 使用注解，一个文件可以处理多个请求，在处理整个项目的请求时，更加轻松，管理起来也更加实用。

* + 1. MyBatis 框架

MyBatis 框架是基于 Java 的持久层框架，用于定制化 sql、存储过程以及高级映射。

原本在 JavaWeb 开发中需要借助 JDBC 工具实现对数据库的 sql 语句操作， 但这样设置 sql 的参数并不方便。MyBatis 消除了几乎所有的 [JDBC](https://baike.baidu.com/item/JDBC) 代码和参数的手工设置以及[结果集](https://baike.baidu.com/item/%E7%BB%93%E6%9E%9C%E9%9B%86)的检索， 使用简单的 XML 或注解用于配置和原始映射， 将接口和 Java 的模型映射成数据库中的记录。

相比传统的持久层操作，MyBatis 框架下的持久层操作更为简单易学。只需要几个 jar 包和几个配置文件将可以成功搭建出所需的环境。搭建完环境后，开发者就可以开始 sql 映射文件的编写。由于 MyBatis 把 sql 语句统一在这些 sql 映射文件中，实际操作中的编写和管理起来也更为方便。

MyBatis 框架的框架架构流程如下：

* + - 1. 加载配置：在开启项目后，服务器会读取我们实现配置的文件以便在后面进行启用。
      2. sql 解析：当前台发送请求调用了事前设定的持久层对象的方法时，会根据方法名称在 sql 映射文件中找出 ID 与之相同的部分，之后传入参数，进行解析，得到最终要执行的 sql 语句和参数。
      3. sql 执行：将之前 sql 解析得到的参数整合到 sql 语句中，然后执行 sql语句。
      4. 结果映射：将操作数据库的结果按照映射的配置进行转换，可以转换成 HashMap、JavaBean 或者基本数据类型，并将最终结果返回。

# 系统设计

系统项目设计思路如下：

## 数据库

Echo 这个项目数据库设计并不复杂，需要我们手动设计的只有四张表：

* 帖子表：discuss\_post
* 评论表：comment
* 用户表：user
* 私信表：message

**用户表**

* id：用户的唯一标识
* username：用户名
* password：存储加盐加密后的密码
* salt：随机生成的盐，用于密码的加盐加密
* email：邮箱
* type：用户类型
  + 0 - 普通用户（用户注册默认是普通用户）
  + 1 - 超级管理员：具有删除帖子、访问数据统计界面的权限
  + 2 - 版主：具有置顶、加精帖子权限
* status：用户状态
  + 0 - 未激活（默认）：用户点击注册后未点击邮箱中的激活链接进行验证，就会处于这个状态。未激活的用户同样无法正常使用某些功能比如发表帖子等
  + 1 - 已激活：用户点击邮箱中的激活链接进行验证成功，就会将状态从未激活改成已激活
* activation\_code：激活码。用户点击注册后，随机生成一串激活码，则在本地环境下：http://localhost:8080/greatecommunity/activation/用户id/激活码 成为该用户的激活链接；在服务器上：http://1.15.127.74/activation/用户id/激活码 成为该用户的激活链接。点击该激活链接则激活用户。激活的逻辑也很简单，就是检查一下这个链接中的用户 id 和激活码是否和数据库中存储的一样。

**帖子表**

* id：帖子的唯一标识
* user\_id：发表该帖子的用户的 id
* title：帖子标题
* content：帖子内容
* type：帖子类型
  + 0 - 普通帖子（默认）
  + 1 - 置顶帖子
* status：帖子状态
  + 0 - 正常（默认）
  + 1 - 精华：为帖子加精可以使其在热度计算中得到一定的加分
  + 2 - 拉黑：管理员删除帖子后，就将这个帖子的状态设置为拉黑
* create\_time：帖子发表时间
* comment\_count：帖子的评论数量（因为会频繁的显示帖子的信息，比如创建时间、创建人、评论数量、点赞数量等，创建时间和创建人信息这张表中已经有了，所以此处再将评论数量存进来就好。可能会有同学会问啥不把点赞数量也缓存到帖子表中，因为点赞数量是存在 Redis 中的，获取点赞数量咱连数据库都不用进的，还费劲在这存一份干啥）
* score：热度 / 分数（用于按照热度排行帖子）

**评论表**

* id：评论/回复的唯一标识
* user\_id：用户 id（哪个用户发布了这个评论/回复）
* entity\_type：实体类型（表示这条 comment 是针对哪个类型的，如果是针对帖子的，那么这个 comment 就是评论；如果是针对评论的，那么这条 comment 就是回复）
* entity\_id：实体的 id（如果是对帖子的评论，就存储帖子的 id；如果是对评论的回复，就存储评论的 id；还有对回复的回复，存储的仍然是所属评论的 id。也就是说，「某个帖子下的所有评论，它们的 entity\_id 都是这个帖子的 id。某条评论下的所有回复，它们的 entity\_id 都是这条评论的 id」）。
* target\_id：目标用户 id（表示这条评论/回复是针对哪个用户的。比如用户 admin 发了一个帖子，用户 master 评论了这个帖子，那么这里的 target\_id 存储的就是用户 admin 的 id。）
* content：评论/回复的内容
* status：评论/回复状态
  + 0 - 正常（默认）
  + 1 - 禁用（暂未使用）
* create\_time：评论/回复发布时间

**私信表**

这张表不仅存储用户之间的私信，也存储系统通知，不同的是，系统通知的 from\_id 特定为 1。用于发送系统通知的角色（用户） SYSTEM 已内置。

字段：

* id：私信/系统通知的唯一标识
* from\_id：私信/系统通知的发送方 id
* to\_id：私信/系统通知的接收方 id
* conversation\_id：标识两个用户之间的对话。比如用户 id 112 给 113 发消息，或者 113 给 112 发消息，这两个会话的 conservation\_id 都是 112\_113。这样，通过这个字段我们就能查出来 112 和 113 之间的私信往来了。当然，这个字段是冗余的，我们可以通过 from\_id 和 to\_id 推演出来，但是有了这个字段方便后面的查询等操作
* content：私信/系统通知的内容
* status：私信/系统通知的状态
  + 0 - 未读（默认）
  + 1 - 已读
  + 2 - 删除（暂未使用）
* create\_time：私信/系统通知的发送时间

## 账号

**注册**

* 用户注册成功，将用户信息存入 MySQL，但此时该用户状态为未激活
* 向用户发送激活邮件，用户点击链接则激活账号（Spring Mail）

**登录|登出**

登录认证模块跳过了 Spring Secuity 自带的认证机制。主要逻辑如下：

* 进入登录界面，随机生成一个字符串来标识这个将要登录的用户，将这个字符串短暂的存入 Cookie（60 秒）；
* 动态生成验证码，并将验证码及标识该用户的字符串短暂存入 Redis（60 秒）；
* 为登录成功（验证用户名、密码、验证码）的用户随机生成登录凭证且设置状态为有效，并将登录凭证及其状态等信息永久存入 Redis，再在 Cookie 中存一份登录凭证；
* 使用拦截器在所有的请求执行之前，从 Cookie 中获取登录凭证，只要 Redis 中该凭证有效并在有效期内，本次请求就会一直持有该用户信息（使用 ThreadLocal 持有用户信息，保证多台服务器上用户的登录状态同步）；
* 勾选记住我，则延长 Cookie 中登录凭证的有效时间；
* 用户登出，将凭证状态设为无效，并更新 Redis 中该登录凭证的相关信息。



**账号设置**

* 修改头像（异步请求）

将用户选择的头像图片文件上传至七牛云服务器

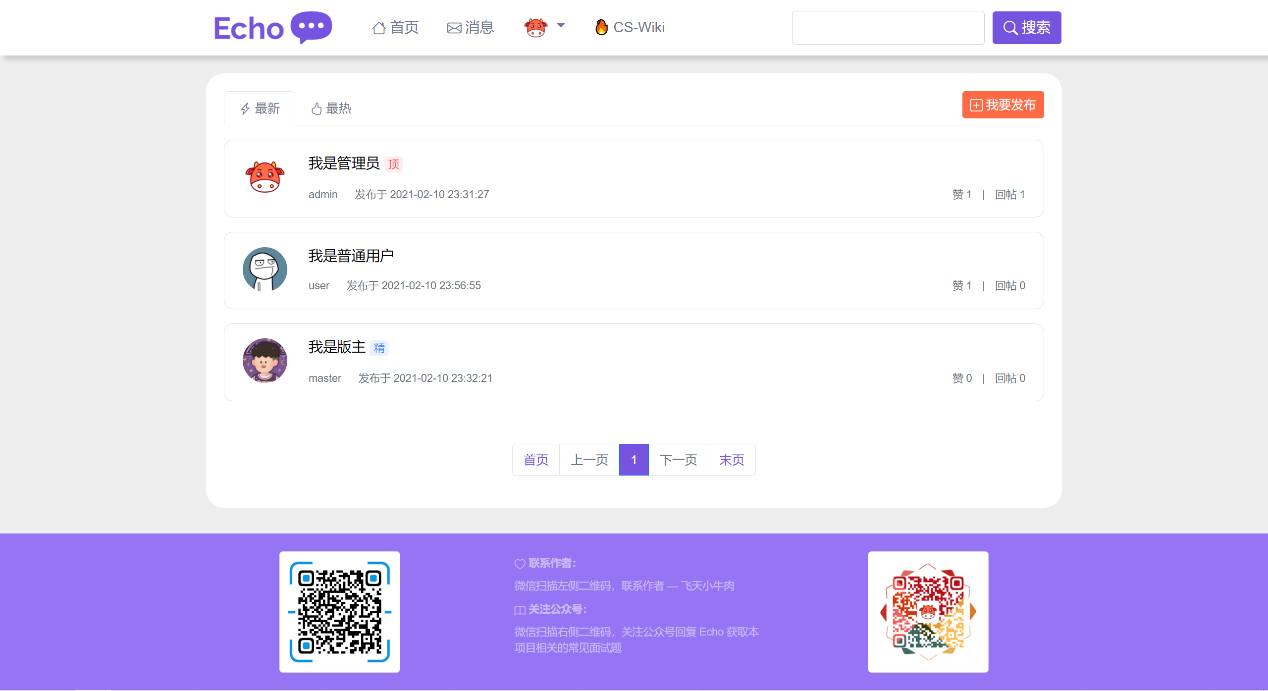
* 修改密码



## 帖子

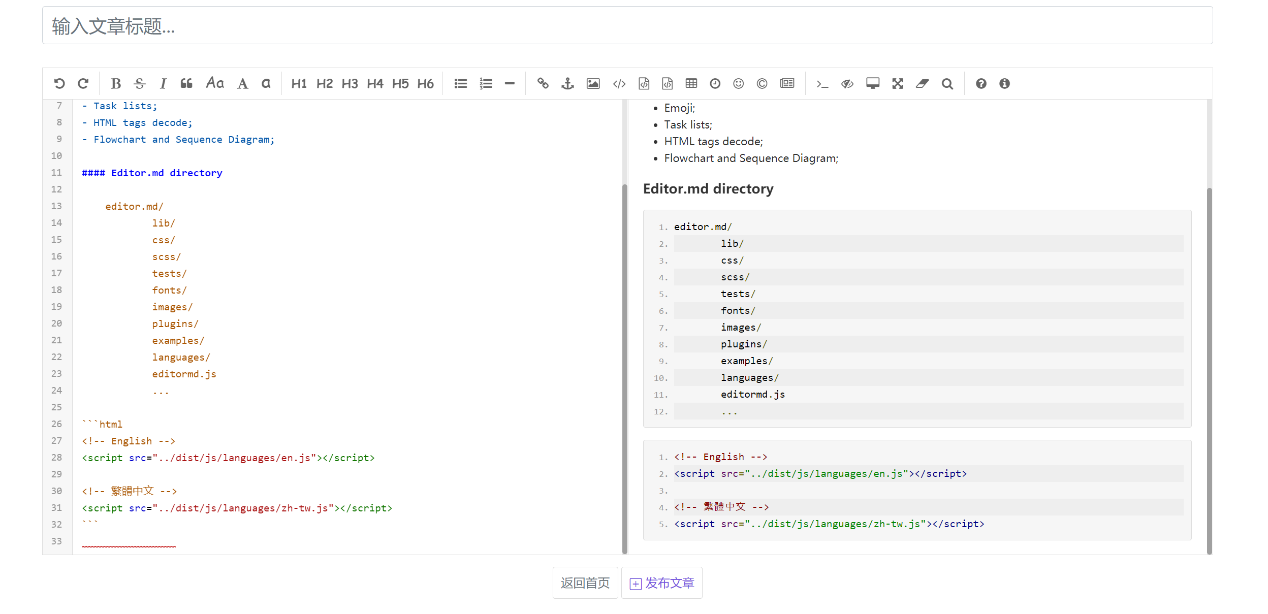
**分页显示所有的帖子**

* 支持按照 “发帖时间” 显示
* 支持按照 “热度排行” 显示（Spring Quartz）
* 将热帖列表和所有帖子的总数存入本地缓存 Caffeine（利用分布式定时任务 Spring Quartz 每隔一段时间就刷新计算帖子的热度/分数 — 见下文，而 Caffeine 里的数据更新不用我们操心，它天生就会自动的更新它拥有的数据，给它一个初始化方法就完事儿）



**发布帖子**

发布帖子（过滤敏感词），将其存入 MySQL



**显示评论及相关信息**

关于评论模块需要注意的就是评论表的设计，把握其中字段的含义，才能透彻了解这个功能的逻辑。

评论 Comment 的目标类型（帖子，评论） entityType 和 entityId 以及对哪个用户进行评论/回复 targetId 是由前端传递给 DiscussPostController 的

**添加评论**

**发布对帖子的评论（过滤敏感词），将其存入 MySQL**

**点赞**

将点赞相关信息存入 Redis 的数据结构 set 中。其中，key 命名为 like:entity:entityType:entityId，value 即点赞用户的 id。比如 key = like:entity:2:246 value = 11 表示用户 11 对实体类型 2 即评论进行了点赞，该评论的 id 是 246

某个用户的获赞数量对应的存储在 Redis 中的 key 是 like:user:userId，value 就是这个用户的获赞数量

**关注**

* 若 A 关注了 B，则 A 是 B 的粉丝 Follower，B 是 A 的目标 Followee
* 关注的目标可以是用户、帖子、题目等，在实现时将这些目标抽象为实体（目前只做了关注用户）

将某个用户关注的实体相关信息存储在 Redis 的数据结构 zset 中：key 是 followee:userId:entityType ，对应的 value 是 zset(entityId, now) ，以关注的时间进行排序。比如说 followee:111:3 对应的value (20, 2020-02-03-xxxx)，表明用户 111 关注了一个类型为 3 的实体即人(用户)，关注的这个实体 id 是 20，关注该实体的时间是 2020-02-03-xxxx

同样的，将某个实体拥有的粉丝相关信息也存储在 Redis 的数据结构 zset 中：key 是 follower:entityType:entityId，对应的 value 是 zset(userId, now)，以关注的时间进行排序

**帖子热度计算**

每次发生点赞（给帖子点赞）、评论（给帖子评论）、加精的时候，就将这些帖子信息存入缓存 Redis 中，然后通过分布式的定时任务 Spring Quartz，每隔一段时间就从缓存中取出这些帖子进行计算分数。

帖子分数/热度计算公式：分数（热度） = 权重 + 发帖距离天数

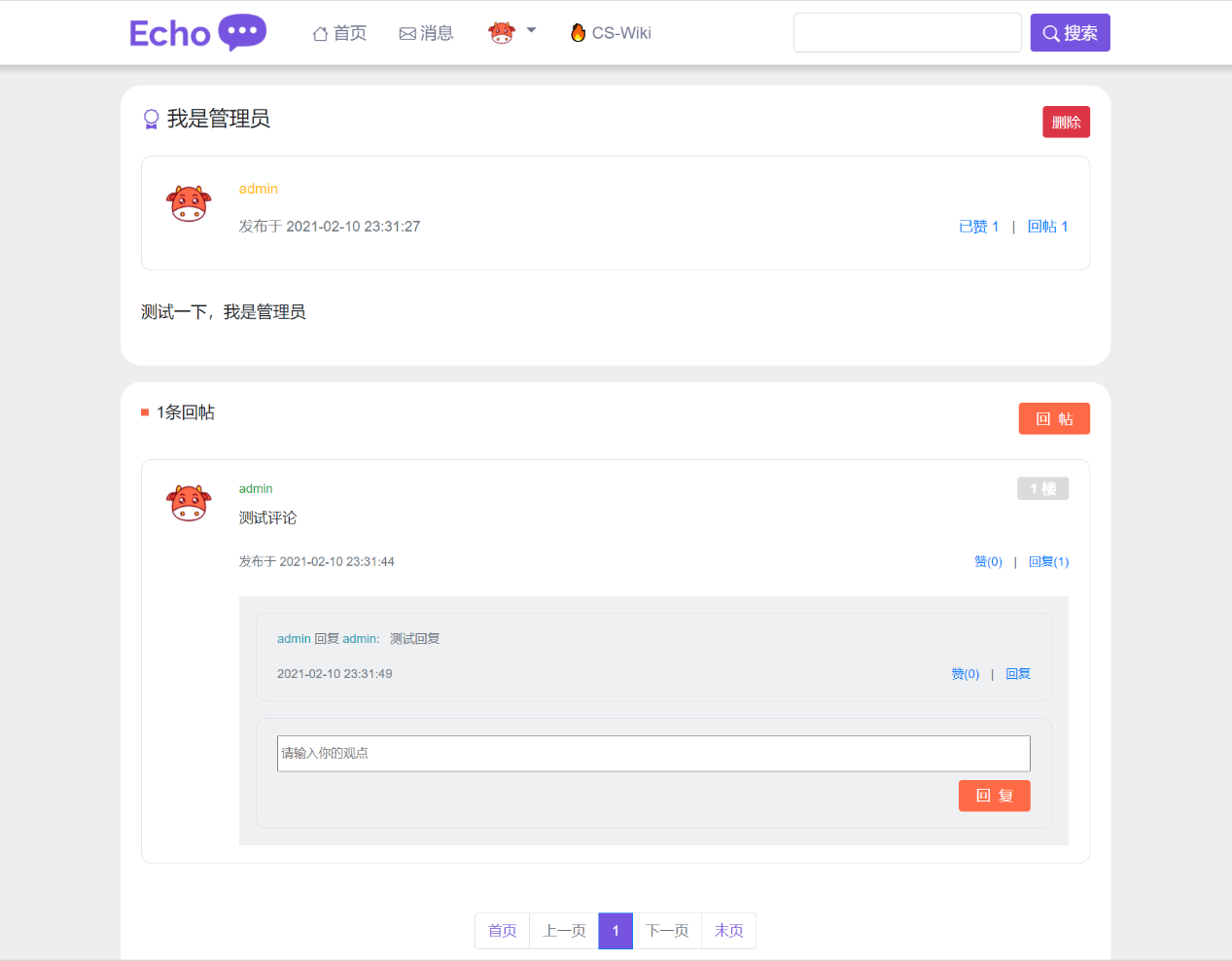
// 计算权重

double w = (wonderful ? 75 : 0) + commentCount \* 10 + likeCount \* 2;

// 分数 = 权重 + 发帖距离天数

double score = Math.log10(Math.max(w, 1))

+ (post.getCreateTime().getTime() - epoch.getTime()) / (1000 \* 3600 \* 24);



## 搜索

* 发布事件

发布帖子时，通过消息队列将帖子异步地提交到 Elasticsearch 服务器

为帖子增加评论时，通过消息队列将帖子异步地提交到 Elasticsearch 服务器

* 搜索服务

从 Elasticsearch 服务器搜索帖子

从 Elasticsearch 服务器删除帖子（当帖子从数据库中被删除时）

* 显示搜索结果



## 数据统计

* 独立访客 UV

存入 Redis 的 HyperLogLog

支持单日查询和区间日期查询

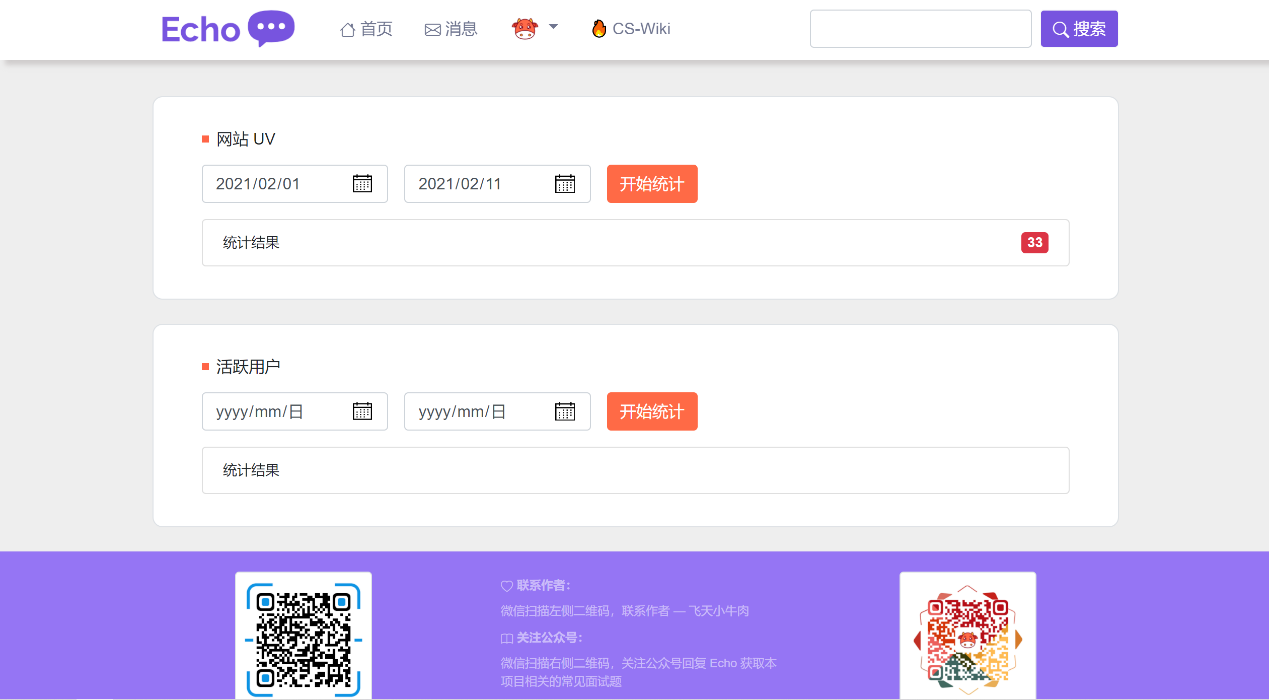
* 日活跃用户 DAU

存入 Redis 的 Bitmap

支持单日查询和区间日期查询

* 权限管理（Spring Security）

只有管理员可以查看网站数据统计



# 总结与展望

在这一门JavaWeb课上，虽然基础不好，但是学习后的收获也非常大。首先要搭建各种环境，由于版本不同，总是报大大小小的错误。关于JavaWeb所要求的掌握的知识点，前端基础3大件、Servlet、Jsp、过滤器、监听器、AJAX、Maven、MyBatis、SpringMVC等等一步步跟着老师学习下来，对JavaWeb的知识架构有了整体性的了解，老师的讲课水平很不错，不拘泥于知识细节，听完课我能对某一个技术有直接宏观的认识。我至今想来都觉得非常震撼的是老师将Maven与Linux的yum仓库类比，虽然没见过用过Maven，但基于我对yum的熟悉，我一瞬间感觉对Maven有了很多很多的了解。

然后，用以上知识做作业，再看书查资料深入了解以上技术实现细节，对这些技术也有很多更加具体的理解和使用经验，做作业不停的报错排错让我没耐心也磨出点耐心了。在以前看到报错，我心里就挺慌的，百度很多解决方法也没有什么用，最后发现往往不是真的有问题，而是某些配置出错导致运行失败报错。现在回看过去，我是有得到锻炼和提高的。

这次写GitHub开源项目的设计文档，GitHub导航站里的一个个开源项目看了之后也有长进了。原本以为自己实力弱，看不懂别人的写好的项目，但为了写文档去仔细看了开源项目后发现，大部分项目是有较为详细的环境搭建与配置，还有简略的内容解说。在写文档的过程中我也是逐渐梳理了一遍这个开源项目的各种内容与细节。虽然说写完文档明白理解这个项目的实现过程，自己做一个JavaWeb项目还是有难度的，但这次作业完成后，让我觉得本来未知的领域我又多探索了一部分。

不论我的技术是否过硬，总体上，这门课的学习成就感还是非常不错的。实现了以及，万事开头难，我已经实现了从无到有的跨越。师傅领进门，修行在个人，感谢老师的引导，唤起我继续学习探索的兴趣。

# 参考文献

[1]百度百科.bootstrap, 2020 -04-10.

https://baike.baidu.com/item/Bootstrap/8301528 .

[2]百度百科.jquery, 2019-10-18.

https://baike.baidu.com/item/jQuery/5385065?fr= Aladdin

[3]简书.2018-08- 30. https://wwwjianshu.com/p/fd7b422d5f93

[4]简书.2018-08-30. https://www.jianshu.com/p/fd7b422d5f93

[5] https://gitee.com/veal98/Echo#https://veal98.github.io/Echo