



华南农业大学

# 本科毕业论文

基于 SSM 的博客网站的设计与实现

邓锦辉

201430320307

指导教师 刘汉兴 讲师

学 院 名 称 数学与信息学院

专 业 名 称 计算机科学与技术

论文提交日期 2018 年 4 月 30 日

论文答辩日期 2018 年 5 月 12 日

## 摘 要

在互联网飞速发展的今天，互联网成为人们快速获取、发布和传递信息的重要渠道。博客是一个在线日记或信息网站，以反向时间顺序显示信息，最新帖子首先出现。这是一个平台，一个作家甚至一个作家团体就个别主题分享他们的观点。越来越多的网民通过博客发表个人的所闻所知，同时也关注他人的生活动态，博客成了信息分享、情感交流的平台。

本网站系统基于先进的MVC思想，采用Spring Boot+MyBatis框架搭建项目，并且集成了现今流行的Spring Cloud微服务框架。用户分为三种身份，分别是游客、普通用户和管理员，身份不同的用户享有不同的权利。

一、游客权利：可查看博主的所有文章和信息，可以给喜欢的文章点赞。

二、普通用户权利：游客可以为自己注册账号，填上注册的用户名、密码、电话和邮箱等信息完成注册。注册成功即可成为普通用户，在登录了系统之后可以查看博主的所有文章和信息，可以给自己喜欢的文章点赞。普通用户可以给文章评论，可以用博主发送消息。

三、管理员权利：管理员登录后可以进入后台管理界面，可以修改博主信息，可以对博文进行增删改查操作，可以管理分类列表，可以增加个人照片或者文章的图片。管理员可以获得普通用户的列表，并且可以对用户进行拉黑操作。

**关键词：**个人博客 SpringBoot SpringCloud Java Mybatis

# Design and Implementation of Personal Blog

Deng Jinhui

(College of Mathematics and Informatics, South China Agricultural University, Guangzhou 510642, China)

**Abstract:** With the rapid development of the Internet, the Internet has become an important channel for people to quickly acquire, publish and transmit information. A blog is an online diary or information website that displays information in reverse chronological order, with the most recent post appearing first. This is a platform where a writer or even a writers group share their opinions on individual topics. More and more netizens publish their personal knowledge through blogs. They also pay attention to the lives of others and blogs have become a platform for information sharing and emotional communication.

The website system is based on advanced MVC ideas, uses the Spring Boot+MyBatis framework to build projects, and integrates the popular Spring Cloud microservice framework. Users are divided into three kinds of identities, namely tourists, ordinary users and administrators. Different users have different rights.

First, the rights of visitors: You can view all bloggers' articles and information, and you can like the articles you like.

Second, ordinary user rights: Visitors can register their own account, fill in the registered user name, password, phone and email to complete the registration. Successful registration can become an ordinary user. After logging in to the system, you can view all articles and information of the blogger, and you can like your favorite articles. Ordinary users can comment on articles and bloggers can send messages.

Third, the administrator rights: administrators can log into the back-end management interface, you can modify the blogger information, you can add, delete, change, check the operation of the post, you can manage the classification list, you can add pictures of personal photos or articles. The administrator can obtain a list of ordinary users and can perform black operations on the users.

**Key words:** Personal Blog   SpringBoot   SpringCloud   Java   Mybatis

# 目 录

1 前言.....	1
1.1 论文选题的背景.....	1
1.2 研究现状以及发展趋势.....	1
1.3 系统开发的意义.....	1
2 框架技术与项目组件.....	2
2.1 MVC 模式分析.....	2
2.1.1 MVC 概念.....	2
2.1.2 MVC 优势.....	2
2.2 Spring Boot 框架.....	3
2.2.1 Spring Boot 背景.....	3
2.2.2 Spring Boot 优点.....	3
2.3 Spring Cloud 微服务框架.....	4
2.3.1 Spring Cloud 简介.....	4
2.3.2 Spring Cloud 优点.....	4
2.4 SpringSession 框架.....	4
2.4.1 Spring Session 简介.....	4
2.4.2 Spring Session 框架优点.....	5
2.5 Nginx 服务器.....	5
2.6 FTP 服务器.....	6
2.7 MyBatis.....	6
2.7.1 MyBatis 简介.....	6
2.7.2 MyBatis 设计特点.....	6
2.7.3 MyBatis 优势.....	7
2.8 Swagger UI.....	7
2.8.1 Swagger 简介.....	7
2.8.2 Swagger 优点.....	7
3 系统分析与设计.....	8
3.1 需求分析.....	8

3.2 系统整体设计.....	9
3.2.1 系统体系结构.....	9
3.2.2 SSM 体系结构.....	11
3.2.3 微服务模块设计.....	13
3.3 数据库设计.....	15
4 系统实现.....	18
4.1 系统开发环境.....	18
4.2 关键技术的实现.....	18
4.2.1 数据源的配置和 ORM 的实现.....	18
4.2.2 Spring Boot 框架搭建.....	22
4.2.3 Spring Cloud 微服务搭建.....	23
4.2.4 Spring Session 框架配置.....	25
4.2.5 Swagger UI 搭建.....	25
4.2.6 AOP 接口鉴权实现.....	27
5 运行效果.....	30
5.1 博客首页.....	30
5.2 文章详情.....	31
5.3 系统管理员.....	34
6 总结与展望.....	36
6.1 总结.....	36
6.2 展望.....	36
参考文献.....	37
致谢.....	38

# 1 前言

## 1.1 论文选题的背景

博客有广泛的应用。个人可以用来撰写博客和发表个人撰写的知识文章。博客是一种深度交流和沟通的新方式。用户可以通过阅读，评论，回复讨论技术知识或分享情感体验，互相学习，共同进步，互相交流讨论。博客还可以用于网络营销，商家可以通过嵌入式广告达到营销目的。

因此，在这个博客流行的时代开发出一个符合大众审美观的博客显得十分重要，博主还需保证内容的正确价值观和知识的正确性。而开发人员则需要保证系统的良好性能。

## 1.2 研究现状以及发展趋势

在当前，博客的在互联网领域的地位绝对占有一席之地。博客技术的发展和运用使得人民受益匪浅，博客的高速发展无疑推动了技术在互联网上的传播，甚至对经济也有一定的影响。许多互联网公司相继出博客服务平台，为的就是拥有一批综合素质和实力高的人才粉丝，这些技术人员能在博客平台上写出高品质的博客文章，从而吸引更多互联网网民浏览博客，使得互联网公司受益。

博客系统在中国拥有广阔的发展前景，互联网公司将个人博客演变成一个博客服务平台，平台上任意一个网民都可以注册一个用户，以此用户名在平台上活动，此平台更加能够促进技术交流、生活分享和正能量传播等等。博客服务平台在此阶段已经显示出功能上的缺乏，创意性的不足，因此未来的博客系统一定是更具有创新性的，只有不断创新，不断完善，博客系统才能更好地服务人民。

## 1.3 系统开发的意义

博客是发布个人信息的一种非常简单的方式。从某种意义上说，它相当于的“博客”。任何人都可以像创建免费电子邮件一样创建，发布和更新个人网页。博客是开放的私人空间，可以及时记录和发布个人工作流程，生活故事，思想史，闪动灵感等。一旦博客研究在关键技术取得新的突破，它将引发前所未有的浪潮在网络世界的博客。更多的人将拥有自己的博客。在这个时候，通过博客的一些新科技将会更容易被推广。政策的好点子也会更容易传播，如果是这样，很明显社会进步会很明显。

## 2 框架技术与项目组件

### 2.1 MVC 模式分析

#### 2.1.1 MVC 概念

模型-视图-控制器（MVC）是一个独立的应用程序架构，分为三种主要的逻辑架构模式：模型，视图和控制器。构建这些组件的每一个都是为了处理应用程序的特定开发方面。MVC 是用于创建可扩展和可扩展项目的最常用的行业标准 Web 开发框架之一(毕建新，2006)。

模型组件对应于用户使用的所有与数据相关的逻辑。这可以表示在 View 和 Controller 组件之间传输的数据或任何其他业务逻辑相关数据。例如，Customer 对象将从数据库检索客户信息，对其进行处理，并将其更新回数据库或用于呈现数据。

View 组件用于应用程序的所有 UI 逻辑。例如，客户视图将包含最终用户与之交互的所有 UI 组件，例如文本框，下拉列表等。

控制器充当模型和视图组件之间的接口，以处理所有业务逻辑和传入请求，使用模型组件处理数据并与视图交互以呈现最终输出。例如，客户控制器将处理客户视图中的所有交互和输入，并使用客户模型更新数据库。同一控制器将用于查看客户数据。

#### 2.1.2 MVC 优势

首先，需要了解 MVC 框架的功能：使用业务逻辑，数据和接口以独立方式显示代码，以将业务逻辑聚合为单个组件。这些部分是独立的和相互关联的。允许每个人都专注于自己的任务（贺松平，2009）。

（1）更快的开发流程：MVC 支持快速的并行开发。使用 MVC，一个程序员可以工作在一个视图上，而另一个程序员可以在控制器上工作，为 Web 应用程序创建业务逻辑。使用 MVC 开发的应用程序比使用其他开发模型开发的应用程序快三倍（杨静，2014）。

（2）在 MVC 模型中，可以为模型创建多个视图。代码重复在 MVC 中非常有限，因为它将数据和业务逻辑从显示中分离出来（张黎明，2007）。

（3）支持异步技术：MVC 还支持异步技术，可帮助开发人员开发非常快速的应用程序。

（4）修改不会影响整个模型：修改不会影响整个模型，因为模型部分不依赖于视图部分。因此，模型中的任何更改都不会影响整个架构。

（5）MVC 模型返回的数据没有格式化：MVC 模式返回的数据没有应用任何格式，

因此可以使用相同的组件并调用任何接口。MVC 模型如图 1 所示：

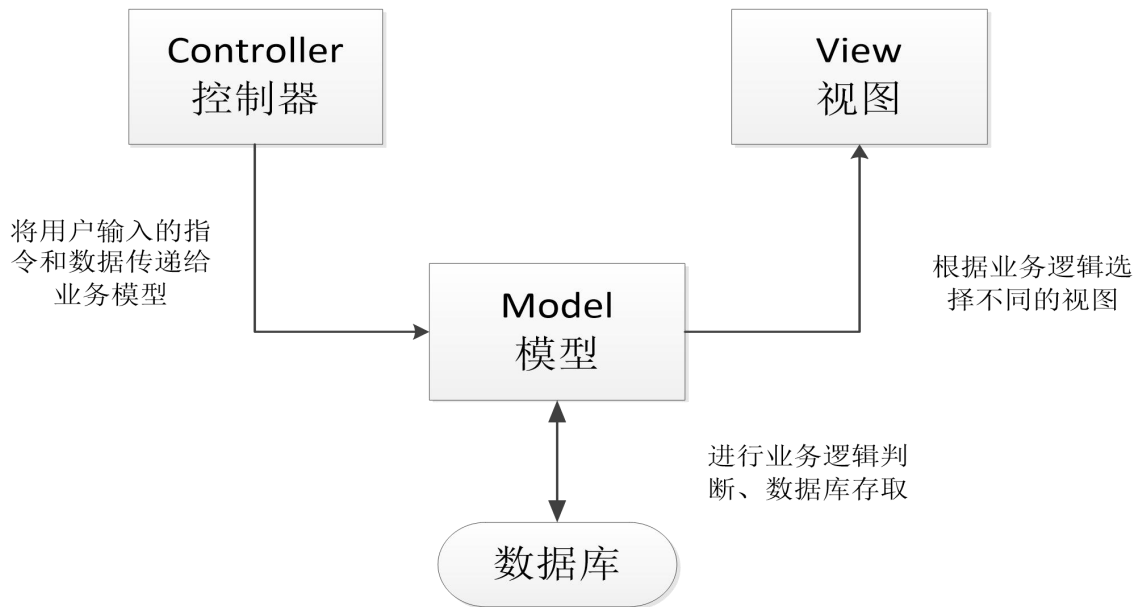


图 1 MVC 模型

## 2.2 Spring Boot 框架

### 2.2.1 Spring Boot 背景

Spring 是用于构建 Web 和企业应用程序的非常流行的基于 Java 的框架。与许多仅关注一个领域的其他框架不同，Spring 框架通过其投资组合项目提供了广泛的功能来满足现代业务需求（俞琰，2009）。

Spring Boot 是一个轻量级框架，可以将大部分工作从配置基于 Spring 的应用程序中解放出来。Spring Boot 的目标是提供一套快速构建易于配置的 Spring 应用程序的工具。Spring Boot 是来自 Pivotal 团队的一个新框架，旨在简化新 Spring 应用程序的引入和开发。框架需要独立的配置方法，无需开发人员定义原型配置。在这方面，Boot 致力于成为不断发展的快速应用程序开发领域的领导者（孙卫琴，2003）。

### 2.2.2 Spring Boot 优点

- （1）通过 Maven 进行简化版和无版本冲突依赖管理。
- （2）可以快速设置和运行独立的 Web 应用程序和微服务。
- （3）Spring Boot 内嵌 Tomcat、Jetty、Undertow 容器，无需自己去配置容器。
- （4）Spring Boot 提供 HTTP 端点来访问应用程序内部信息，如详细指标、应用程序内部工作、健康状况等。



(5) 没有基于 XML 的配置，简化了属性，通常通过代码配置 bean。

(6) Spring 初始化程序提供了一个项目生成器，允许从一开始就使用特定的技术堆栈。可以使用 Web，数据访问（关系和 NoSQL 数据存储），Spring Cloud 或消息传递支持来创建框架项目。

(7) Spring Boot 默认使用内嵌的 Tomcat，默认端口是 8080，如果要修改 tomcat 的端口非常简单，只需要在 application 文件中添加代码如图 2 所示：



图 2 Application 文件

## 2.3 Spring Cloud 微服务框架

### 2.3.1 Spring Cloud 简介

Spring Cloud 为开发人员提供了快速构建分布式系统中一些常见模式的工具（例如配置管理，服务发现，断路器，智能路由，微代理，控制总线，一次性令牌，全局锁定，领导选举，分布式会话，群集状态）。分布式系统的协调导致锅炉板模式，使用 Spring Cloud 开发人员可以快速站出实现这些模式的服务和应用程序。

### 2.3.2 Spring Cloud 优点

Spring Cloud 微服务可以独立部署，它不像传统的单体架构，单体架构只要有一处地方出错整个项目就运行不起来，因此排查错误成了一个非常大的难题；而微服务架构将一个大的应用程序通过粗细度拆分成一个个小型服务程序，每一个服务都可以独立运行，可以拥有自己的数据库，也可以使用不同的语言编写。这些微服务可以独立部署，不会互相依赖，某个服务出错不影响其他服务启动，因此排查错误的效率大大增加。它的灵活性体现在不像传统单体架构只能够垂直集群分布，微服务可以只将某个服务集群，使服务器的资源使用更加高效。Spring Cloud 微服务能够将资源有效地隔离，每一个微服务拥有独立的数据源，这样有效避免了服务之间争用数据库和缓存资源所带来的问题。

## 2.4 SpringSession 框架

### 2.4.1 Spring Session 简介

在分布式系统中，必须解决的一个重要问题是如何统一服务器的会话信息。当正在做项目的群集时，会话总是需要解决的问题。过去可以从 Servlet 容器中解决它，或者像 Nginx 一样使用 ip\_hash 来路由到特定的服务器。但是，这两种方法都有缺点。过度依赖 Servlet 容器很容易导致问题。Spring Session 提供了用于管理用户会话信息的 API 和实现。SpringSession 使得支持集群会话的过程变得非常简单，而不会受限于特定于应用程序容器的解决方案。它还提供透明的整合，允许替换 HttpSession 应用程序容器（即 Tomcat）中立的方式，支持在头中提供会话 ID 以使用 RESTful API。spring-session 是 spring 旗下的一个项目，把 servlet 容器实现的 HttpSession 替换为 spring-session，专注于解决 session 管理问题。可简单快速且无缝的集成到的应用中（陈雄华，2009）。

### 2.4.2 Spring Session 框架优点

Spring Boot 可以无缝集成 Spring Session+Redis，需要配置相应的 Maven 依赖，配置类需继承 HttpServletRequestWrapper、HttpServletResponseWrapper 两个类，在 application 文件中配置 Redis 的连接信息，启动程序前需要启动 Redis 服务器。在完成这几个简单的操作之后 Spring Session 就起作用，每次请求到达后端时 HttpSession 都会被 Spring Session 自动生成的 Session 替换掉，进而保存在配置好的 Redis 服务器中。

此方法操作简单，并且无需依赖 Servlet 容器，就能够实现分布式系统的 Session 信息统一，解决了分布式系统中 Session 统一的难题，前提是 Redis 容量足够大，并且需要一直运行（Azat Mardan，2014）。

## 2.5 Nginx 服务器

Nginx 是一款轻量级 Web 服务器，专为反向代理，缓存和负载平衡而设计。它最初是为最大的性能和稳定性而设计的。Nginx 背后的目标是创建最快的 Web 服务器，而保持卓越仍然是该项目的核心目标。

在这个项目中使用了 Nginx 的核心功能之一：反向代理。它允许 Nginx 将请求传递到后端 http 服务器进行进一步处理。Nginx 通常设置为反向代理解决方案，以帮助扩展基础架构或将请求传递给其他服务器，这些服务器不是为处理大型客户端负载而设计的。从 Nginx 代理其他服务器的理由是能够扩展的基础架构。Nginx 可以同时处理多个并发连接。这使得它成为客户联系人的理想选择。服务器可以向任意数量的后端服务器来处理大量工作并将负载分散到基础架构中。

## 2.6 FTP 服务器

FTP 着手解决发布文档和软件的需求，以便人们可以轻松地从一个计算机系统获取它们。在 FTP 服务器上，文件按目录结构组织。用户可以通过网络连接到服务器，在目录结构中上下移动以找到他们感兴趣的文件，并从服务器下载文件。

最初 FTP 服务器的一个缺点是，当人们在 Internet 上查找文件或文档时，他们必须知道哪个 FTP 服务器保留了他们正在查找的文件。然而，随着网络的出现，用户现在可以依靠网页上的各种搜索引擎和链接来帮助识别他们想要的文件的 FTP 服务器。实际上，当下载通过点击网页中的链接，甚至可能不知道该文件是从 FTP 服务器下载的。FTP 服务器使用 FTP 协议传输文件和图片等文件。基于 TCP 的原理，它是一个安全的，全双工的，可靠的服务器，是 Web 开发的好帮手。

## 2.7 MyBatis

### 2.7.1 MyBatis 简介

MyBatis 是一个优秀的持久性框架，支持自定义 SQL，存储过程和高级映射。MyBatis 可以避免几乎所有的 JDBC 代码并手动设置参数并获取结果集。MyBatis 可以使用简单的 XML 或注释来配置和映射本机信息，将接口和 Java 的 POJO（普通旧 Java 对象）映射到数据库中的记录（谢世波，2003）。

MyBatis 是一个开源的轻量级持久性框架。它是 JDBC 和 Hibernate 的替代品。它可以自动执行 SQL 数据库与 Java，.NET 和 Ruby on Rails 中的对象之间的映射。通过将 SQL 语句打包到 XML 配置文件中，映射与应用程序逻辑分离（贾文潇，2016）。

它抽象几乎所有 JDBC 代码，并减轻手动设置参数和检索结果的负担。它提供了一个简单的 API 来与数据库进行交互。它还提供对定制 SQL，存储过程和高级映射的支持。它以前称为 IBATIS，由 Clinton Begin 于 2002 年创建。MyBatis 3 是最新版本。这是 IBATIS 的完整转型。

### 2.7.2 MyBatis 设计特点

- （1）简单- MyBatis 被广泛认为是当今最简单的持久性框架之一。
- （2）可移植性- MyBatis 几乎可以用于任何语言或平台，例如用于 Microsoft .NET 的 Java，Ruby 和 C#。
- （3）独立接口- MyBatis 提供独立于数据库的接口和 API，帮助应用程序的其余部分保持独立于任何与持久性相关的资源。
- （4）开源- MyBatis 是一个开源软件，不用付费，在 github 上可找到。

### 2.7.3 MyBatis 优势

- (1) 支持存储过程 - MyBatis 以存储过程的形式封装 SQL，以便业务逻辑可以不在数据库中，而且应用程序更具可移植性，更易于部署和测试。
- (2) 支持内联 SQL - 不需要预编译器，可以完全访问 SQL 的所有功能。
- (3) 支持动态 SQL - MyBatis 提供了基于参数动态构建 SQL 查询的功能。
- (4) 支持 O/RM - MyBatis 支持许多与 O/RM 工具相同的功能，例如延迟加载，连接提取，缓存，运行时代码生成和继承。

## 2.8 Swagger UI

### 2.8.1 Swagger 简介

Swagger 基本上是一组用于语义描述 API 的规则（规范）和工具。事实上，这是一个独立于语言的工具，它允许每个人都在同一页面上。Wagger 是一个用大家都能理解的通用语言来描述的 API 的框架。

### 2.8.2 Swagger 优点

- (1) 这对开发人员和非开发人员来说很容易理解。产品经理，合作伙伴甚至潜在客户都可以为 API 的设计做出贡献，因为他们可以在这个友好的用户界面中清楚地看到它。
- (2) 它是人类可读和机器可读的。这意味着，不仅可以在内部与的团队共享，还可以使用相同的文档来自动化依赖于 API 的流程。
- (3) 它很容易调整，这对于测试和调试 API 问题非常有用，首页如图 3 所示：

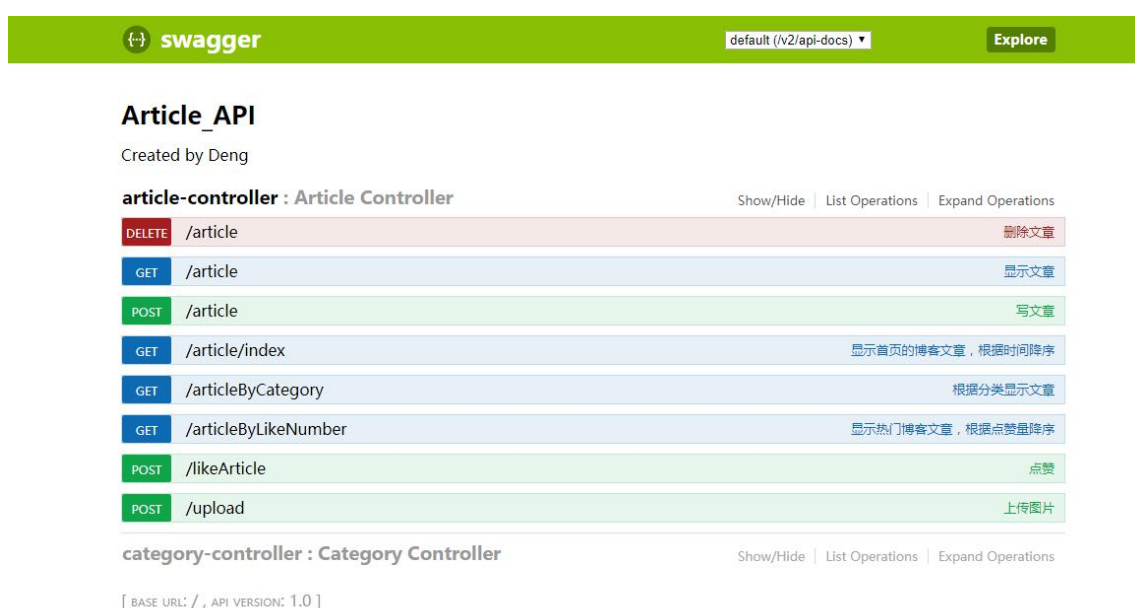


图 3 Swagger UI 主界面

### 3 系统分析与设计

#### 3.1 需求分析

经过合理的分析与设计，本博客系统分为前台和后台。前台的使用者是普通用户，而后台的使用者是管理员，即个人博客的博主，下面将会针对前后台进行详解。

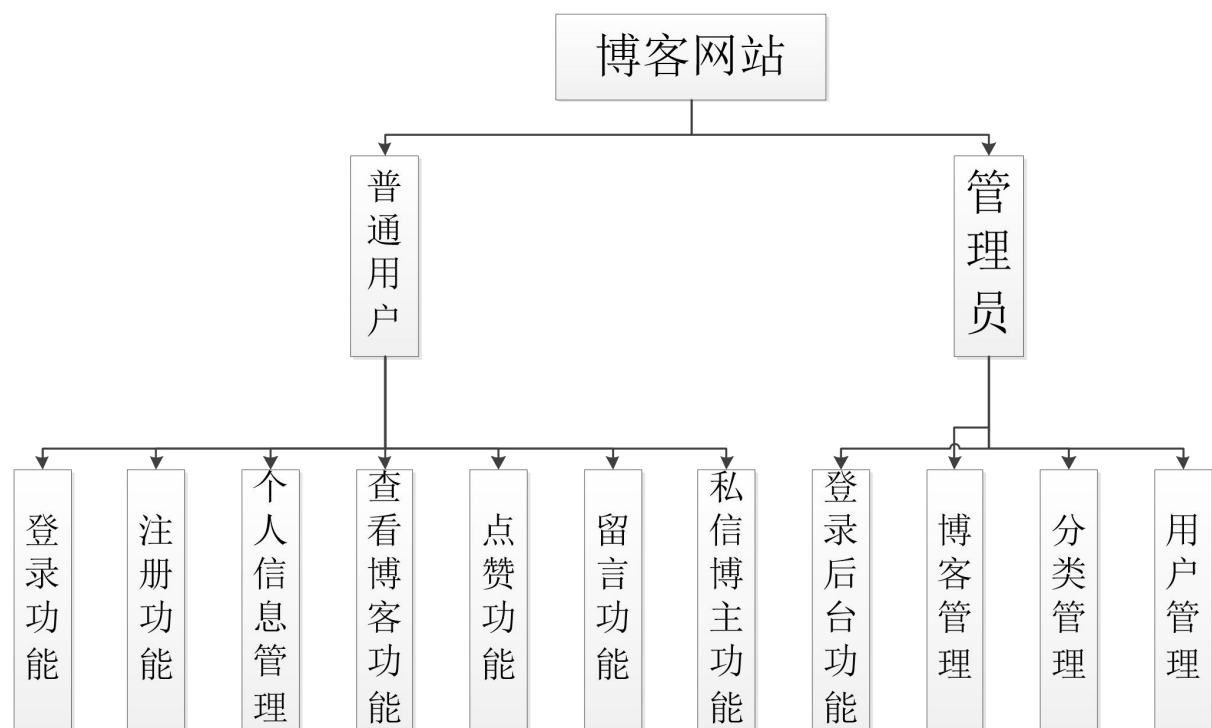


图 4 系统模块图

前台即普通用户使用系统的范围，图中的左半部分的功能都是前台的功能模块，普通用户即非管理员的所有用户。

（1）注册功能，没有账号的普通用户只能查看博主的博客文章，并不能在文章底下留言，也不能在私信博主，因此注册功能是享有这些功能的入口。

（2）登录功能，当输入正确的用户名和密码即可登录，系统保存登录的状态信息。

（3）个人信息管理，用户的个人信息默认为注册时填写的个人信息，该功能允许用户修改自己的个人信息、查看他人的个人信息。

（4）查看博客功能，所有用户（包括网民）可以查看博主的所有博客。博客首页根据博客的点赞量进行倒序排序，用户也可以通过点击不同的分类查看各分类列表中的文章。

(5) 点赞功能，当用户觉得该文章是篇好文章，可以点赞文章，增加文章的热度。

(6) 留言功能，用户可以文章底下留言，对文章进行评价，或者与他人进行讨论。该功能包含了回复功能。

(7) 私信博主功能，登录后的用户可以向博主发私信。

后台即管理员使用系统的范围，图中的右半部分的功能是后台的功能模块，管理员即个人博客的博主。

(1) 登录后台，管理员也拥有账号和密码，当输入的账号和密码验证为管理员即跳转到后台管理页面。

(2) 博客管理，管理员拥有博客的管理所有权，允许对博客进行增删查改操作。

(3) 分类管理，管理员拥有分类的管理所有权，允许对分类进行增删查改操作。

(4) 用户管理，管理员可以看到所有用户的信息，也可以将某个用户拉进黑名单或者删除。

## 3.2 系统整体设计

### 3.2.1 系统体系结构

本博客系统的后端以 Spring+SpringMVC+MyBatis 作为架构基础，Spring Cloud 微服务架构作为延伸，搭建了现今流行的 Spring Cloud 微服务系统。Spring Boot 本着习惯优于配置的思想，省去繁琐的 XML 文件配置，加快了 SSM 框架的搭建和业务的开发。而且 Spring Boot 开发出来的系统无需打包成 war 包，只需打包成 jar，发布到任何装有 java 虚拟机的系统即可启动。Spring Boot 内置了 Tomcat 容器，无需自己搭配 Tomcat 服务器。Spring Boot 还无缝集成了许多优秀的框架，如 Spring Session、Spring Security、RabbitMQ 等等。同时 Spring Boot+MyBatis 是基于 SSM 的，因此本系统使用 Spring Boot 进行开发。

Spring Cloud 微服务框架能够快速开发一个分布式系统。Web 系统可能出现以下几个因素影响系统的性能，分别是服务器的性能、网络的带宽、数据库的优化不足或者 SQL 语句低效、客户的并发访问增大和单体系统的瓶颈等等。任何的 Web 系统一旦业务增多，客户访问量增大，网站系统的并发量都会增大，由于单体系统在单台服务器上部署，服务器会不堪负重，此时应该给系统做集群或者分布式。因此，要开发一个分布式系统，Spring Cloud 是一个好的选择。系统具体架构如图 5 所示：

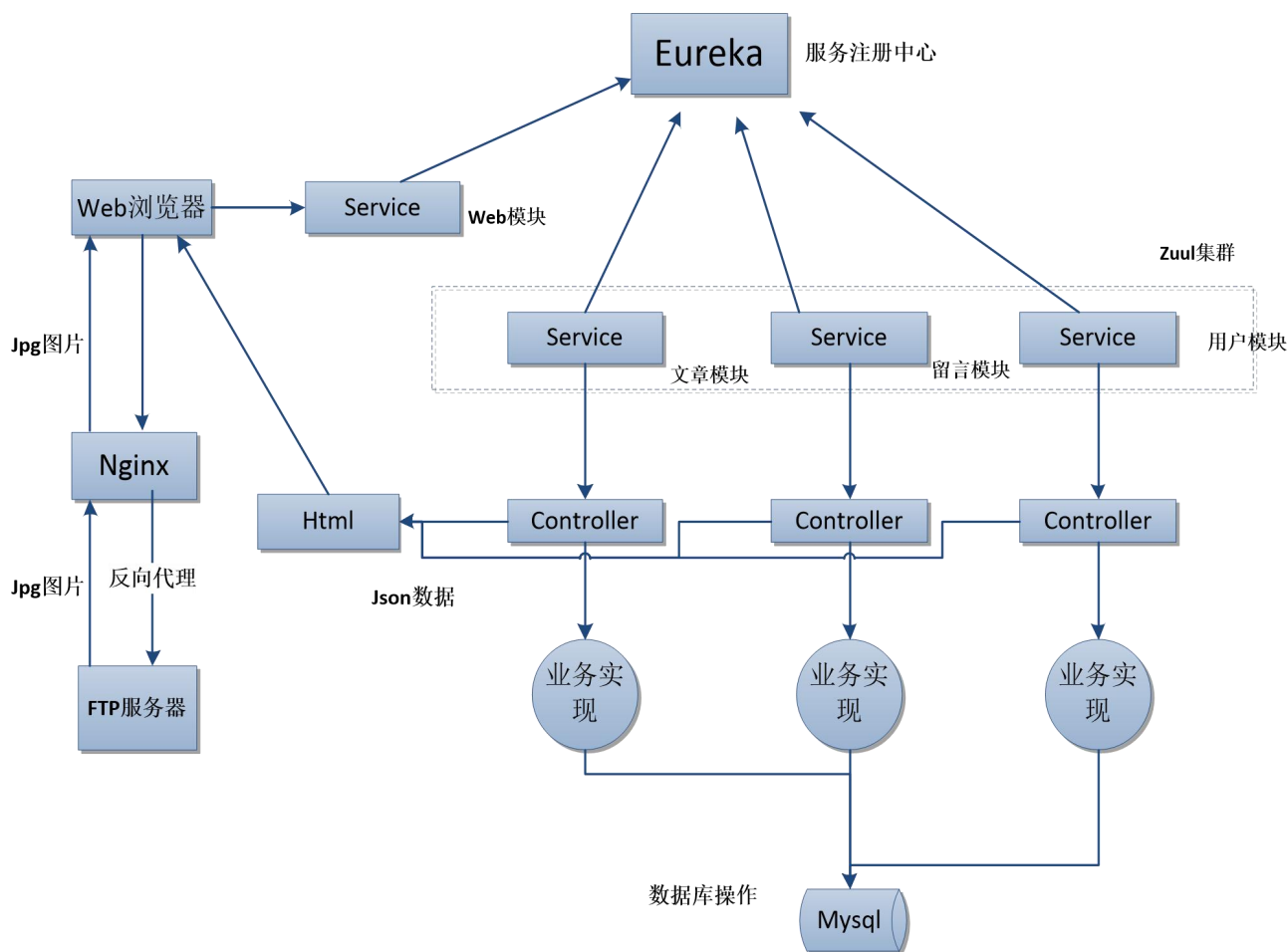


图 5 系统架构图

系统的总体架构如图 5 所示, 下面将详细介绍架构的各组件与组件的作用。

(1) **Eureka**: 微服务需要一个服务注册中心，而 Eureka 组件正式 Spring Cloud 的服务注册中心。它相当于服务器，用于记录服务的注册与发现。当服务向其注册之后，Eureka 将接收服务发送的心跳包已检测该服务是否出现故障或终止，当接收不到心跳包后则在服务列表将该服务注销。Eureka 支持集群，数据同步可以在集群中的 Eureka 实例之间执行。所有 Eureka 实例中的服务注册信息都是同步的。

(2) **Service**: 即一个完整的服务，一个合格的微服务系统根据系统的功能合理地划分为不同服务，每一个服务都拥有自己的 Tomcat 容器，拥有自己的资源，可以独立地运行。服务与服务之间通过 Rest 调用，可以看作是一个小型的项目。Service 通过合理的配置后运行项目，即可向 Eureka 注册。

(3) Zuul 集群：即可通过 Zuul 路由访问的服务。在该系统中浏览器永远都是访问 Web 模块的 URL，Web 模块中配置了 Zuul，将不同的 url 路由到不同的服务。具体的业

务逻辑实现则在各模块中执行。

(4) **Controller**: 控制器, Spring MVC 的核心, 客户的请求都会根据 Controller 的 Mapping 被映射到不同控制器, 控制器再调用 Service 接口层完成业务的实现。业务的实现最终向控制器返回 Json 数据, 控制器再向前端返回 Json 数据。

(5) **FTP 服务器**: 文件服务器, 可以对文件进行上传和下载。在该项目中用于图片的存取, 管理员写新的博客时可上传图片, 图片则会上传到 FTP 服务器。

(6) **Nginx 服务器**: 一款优秀的、轻量级的 Web 服务器, 也可用作反向代理器和负载均衡器。在该项目中主要用于反向代理获得 FTP 服务器的图片。因为 FTP 的图片需要 FTP 协议才可以获取, 所以前端的 HTTP 协议无法获取, 此时, 使用 Nginx 服务器反向代理 FTP 服务器的文件夹, 即可获取图片 (Peter Ljungstrand, 2000)。

### 3.2.2 SSM 体系结构

本系统基于 SSM 框架, 即 Spring+Spring MVC+MyBatis, SSM 框架是当下流行的 Web 开发框架, 受到 Java 开发工程师的青睐。下面将详细介绍 SSM 框架以及它的优点。

(1) **Spring** 是一个开源的设计级框架。它解决了业务逻辑层和其他层之间松散耦合的问题。因此, 它将在整个系统中使用面向接口的编程概念。可以这样说, Spring 颠覆了对编程的一些传统观念。Spring 最大的两个特点分别是 IOC 和 AOP。

① **IOC**: 中文为控制反转, 它的意思是 Spring 容器将传统地在代码中对对象的创建和依赖的操作交给它去完成, 不需要自己 new 一个对象出来对它管理, 这样做的好处是只需要通过配置文件或者注解的方式对对象创建和依赖注入, 不让这些操作出现在的代码中, 不同类高度解耦, 提高维护性 (Bruce Eckel, 2006)。

② **AOP**: 面向切面编程, 与面向对象编程互补, 弥补了面向对象编程的缺点。一般的编程都是自上而下, 运行也是自上而下。而 AOP 定义了一个切面, 这个切面是一个类, 里面封装了与业务逻辑无关的代码, 封装了公共的行为。AOP 就像一把刀, 在横切点进行一刀切, 就像将多个类的代码切开, 然后向其中填补切面的公共行为。AOP 的业务场景有接口鉴权、异常处理、打印日志和测试性能等等 (Craig Walls, 2006)。

(2) **Spring MVC** 是一个基于 MVC 的 Web 框架。Spring MVC 分离了控制器, 模型对象, 过滤器和处理程序对象的角色。这种分离使他们更容易定制。Spring MVC 属于 SpringFrameWork 的后续产品, 已经融合在 Spring Web Flow 里面。

① Spring MVC 执行流程如图 6 所示:



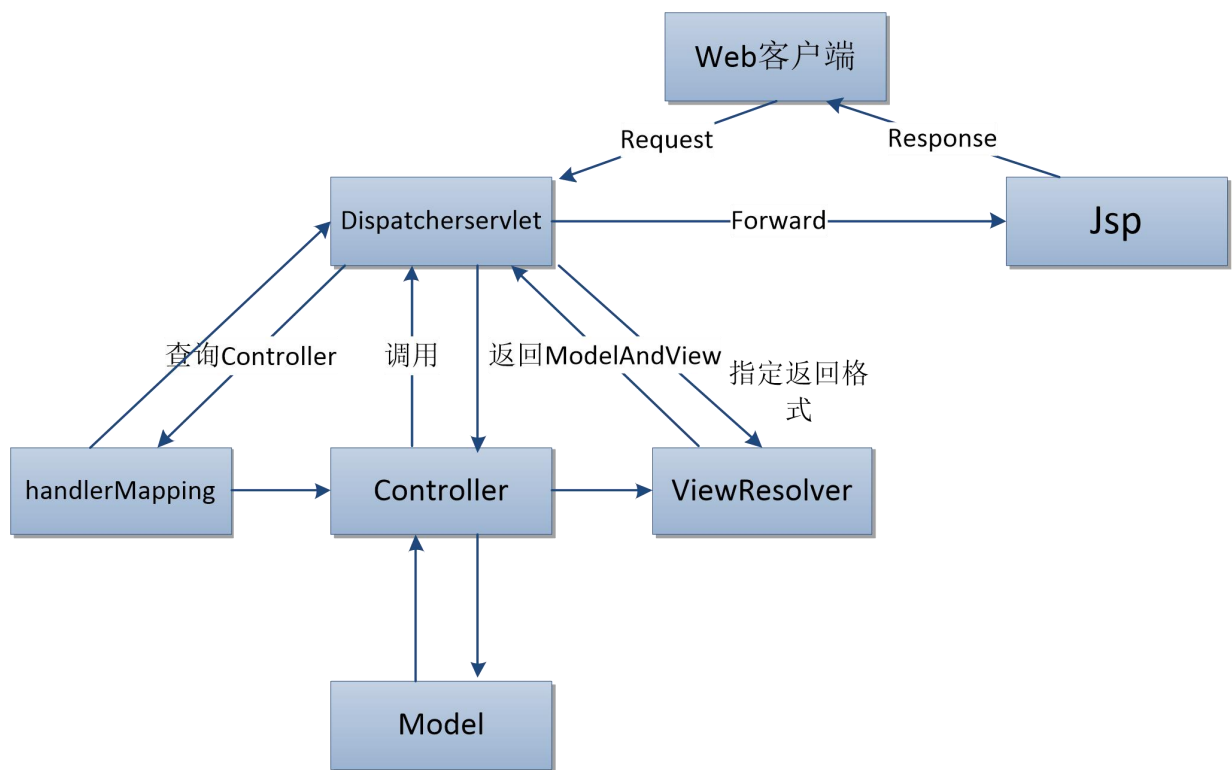


图 6 Spring MVC 流程图

客户发送请求之后，请求到达 DispatcherServlet 前端控制器。DispatcherServlet 会调用 HandlerMapping，并且与请求的 URL 匹配，找到相应的 Handler，此时 HandlerAdapter 会对 Handler 封装，并对所有 Handler 调用统一的接口，相应的 Controller 执行请求。Controller 执行完返回 ModelAndView 给 DispatcherServlet，DispatcherServlet 调用视图解析器对模型进行解析，解析完后向 DispatcherServlet 返回 View，DispatcherServlet 对对象模型进行渲染，渲染完毕向前端返回相应的文件，这个文件可能是 HTML、图片或者文件等等。

② Spring MVC 有很多优点。Spring MVC 使用一组 MVC 注释来使 POJO 成为一个无需实现任何接口即可处理请求的控制器。它使用松散耦合的可插拔组件结构，比其他 MVC 框架更具可扩展性和灵活性，使得 Web 层的开发更加简洁。Spring MVC 与 Spring 框架（例如 IoC 容器，AOP 等）固有集成，可以轻松地对 Web 层执行单元测试，并支持 RESTful URL 请求，并支持灵活地将 URL 映射到页面控制器。而且，Spring MVC 非常容易与其他视图技术（如 Velocity，FreeMarker 等）集成。

（3）MyBatis: MyBatis 是一个数据持久层（ORM）框架，它在数据库表与实体类之间建立映射关系，当想对数据库中的数据操作时只需要对实体类进行操作，这样的思想符合面向对象思想。SQL 语句都定义在 XML 文件中，这些映射文件称之 Mapper，一

个 XML 文件对应一个 Mapper 接口类，该类向外提供 API 接口，这些接口对应着 XML 文件中定义的方法，便于统一管理和维护，降低了程序的耦合度。Hibernate 不同于 MyBatis 的一个地方是 Hibernate 不用自己编写 SQL 语句，这算是一个优点，减少开发难度，但同时也是一个缺点，因为不编写 SQL 语句意味着调优问题得不到解决。而 MyBatis 可以任意编写自己的 SQL 语句，高级程序员懂得如何在 SQL 角度上调优系统，这是 MyBatis 的一个优点，因此系统也使用了 MyBatis 作为 ORM 框架。

### 3.2.3 微服务模块设计

(1) **Server** 模块作为服务注册中心，该模块没有业务逻辑，纯作为一个服务端提供服务注册，可以复制几份 Server 代码，改变 Tomcat 的端口，并且在配置文件中配置集群的相关信息，即可实现一个高可用的服务注册中心。

(2) **Web** 模块作为客户端的访问入口，该模块没有业务逻辑，浏览器访问系统的任何功能都是在基础该模块的操作。该模块存放着所有的静态资源如 HTML、CSS、JS 文件等等。该模块中配置了 Zuul 路由，访问该模块配置的路径即可路由到其他模块的控制器实现业务逻辑。

(3) **Customer** 模块为用户模块，用户模块的两个核心功能是注册和登录。注册时，先检查是否已存在该用户名，如果存在即返回相关信息；如果不存在的话即可开始注册的业务逻辑实现。系统会将用户输入的密码用 MD5 明文加密方式给密码加密，加密后连管理员也无法从数据库表中得知密码。当所有信息验证无误后即向数据库中的用户表添加一条记录，当 DAO 层返回添加成功后即注册成功，向前端返回注册成功信息。登录时，系统先检查是否已存在该用户名。若不存在，则向前端返回错误信息；若存在，则将密码进行 MD5 明文加密，再到数据库中的用户表查询是否存在满足用户名和密码都对应的记录，若存在该记录，即登录成功，向前端返回登录成功的信息，否则返回密码错误。用户模块还有删除用户和修改用户功能，修改用户功能即用户修改个人信息功能，删除用户功能供管理员使用，管理员有权利将用户删除。

(4) **Message** 模块为留言模块，留言模块有三个核心功能，分别是留言、回复留言和根据文章 ID 查找留言。留言功能和回复留言功能都必须在用户已登录的状态下才能使用，否则将跳到登录的页面。用户发送留言请求时，系统先访问一次数据库看存在多少记录，即可判断当前是第几楼，然后填充留言的 bean 信息，向数据库写入这条留言。回复留言之时，前端向用户传被回复的用户的信息，然后系统访问一次数据库看存在多少记录，即可判断当前是第几楼，然后填充留言的 bean 信息，向数据库写入这条留言。根

据文章 ID 查找该文章的留言的功能作为 API 接口给文章模块调用，因为文章模块显示文章信息时也需要找到该文章的留言情况。

(5) **Article** 模块为文章模块，包括了文章管理和分类管理。分类管理中包含着对分类的增删查改操作。文章管理中包含着对文章的增删查改操作，其中查操作中还分为首页查找、根据分类查找、根据点赞量倒序查询。通过文章的 ID 查询文章的详情，同时阅读量加一。该模块中还包含着一个远端接口的调动，该接口的作用是查询某篇文章的留言详情。该接口使用 **Feign** 客户端配置，**Message** 模块作为生产者，本模块作为消费者，此模式为 **Spring Cloud** 各服务中典型的互相调用方式。

(6) **Common** 模块为公共模块，该模块不作为一个服务，而是作为一个公共模块让其他模块共用。其不包含业务逻辑，包含的都是每个服务都要用到的公共行为。它包含着如下组件：

① 自定义注解：本系统自定义了一个注解 **Check**，该注解用于注解方法，搭配 **AOP** 实现接口鉴权。因为有些接口如查询首页博客是不需要登录的，即不需要接口鉴权。而部分接口如评论、回复评论等接口是需要用户登录后才能实现业务逻辑的，这些接口就需要接口鉴权，本系统在需要鉴权的接口方法上使用该注解。

② 鉴权 **AOP**：使用 **AOP** 的方式对需要验证权限的接口进行鉴权。**Pointcut** 定义为自定义的注解 **Check**。使用 **Around** 注解对方法进行 **AOP**，首先拿到 **HttpServletRequest** 的 **session**（该 **session** 通过 **Spring Session** 特殊处理过），再通过 **joinPoint** 拿到“**userId**”这个参数名，判断 **session** 中是否存在对应的数据，如果为空，表示还没登录，返回错误信息。如果已登录，则放行。

③ **DataSourceConfig**:数据库的数据源。

④ 自定义运行时异常和 **ControllerAdvice**：自定义运行时异常用于表示某些业务逻辑的错误。**ControllerAdvice** 用于在 **Controller** 层全局捕获异常，这样就不用业务代码中添加 **try-catch** 这些不必要的代码，让代码变得更加简洁，也让异常处理拥有更高的维护性。

⑤ 高复用返回类：该类包含着状态码、信息和数据，用于服务端向前端返回的类。

⑥ **Utils** 包：该包包含 **FTPUtil** 和 **MD5Util**，前者用于 **FTP** 服务器对文件的上传和下载，后者用于用户密码的加密方式。

⑦ **Const** 常量：包含着所有服务可能用到的常量，统一管理常量提高维护性。

### 3.3 数据库设计

数据库有多种，也分关系型与非关系型数据库，选择一个合适的数据库是数据处理的关键。由于 Mysql 语法易学易懂，性能良好，而且是关系型数据库，适合开发中小型系统，因此本系统使用 Mysql 数据库。

用户表包含了用户的个人信息，具体如表 1 所示：

表 1 用户表 user

字段	字段名	类型	宽度	是否主键	是否为空
ID	用户编号	char	40	是	Not null
USERNAME	用户名	char	32	否	Not null
PASSWORD	用户密码	char	32	否	Not null
PROFESSION	职业	char	32	否	null
PHONE	电话	char	32	否	null
EMAIL	电子邮箱	char	32	否	null
SEX	性别	int	1	否	null
CREATTIME	创建时间	datetime	无	否	Not null

留言表保存的是文章的留言信息，具体如表 2 所示：

表 2 留言信息表 message

字段	字段名	类型	宽度	是否主键	是否为空
ID	留言编号	char	40	是	Not null
FLOOR	楼层编号	int	11	否	Not null
ARTICLE_ID	文章编号	char	40	否	Not null
CONTENT	内容	text	无	否	Not null
UPDATED_TIME	留言时间	datetime	无	否	Not null
UPDATED_USER	用户姓名	char	20	否	Not null

文章信息表包含文章的基本信息，具体如表 3 所示：

表 3 文章信息表 article

字段	字段名	类型	宽度	是否主键	是否为空
ID	文章编号	char	40	是	Not null
CATEGORY_ID	分类编号	char	40	否	Not null
INTRODUCTION	简介	text	无	否	null
CONTENT	内容	text	无	否	Not null
PICTURE	图片 URL	char	200	否	null
TITLE	标题	char	40	否	Not null
READ_NUMBER	阅读量	int	11	否	Not null
LIKE_NUMBER	点赞量	int	11	否	Not null
MESS_NUMBER	留言量	int	11	否	Not null
UPDATED_TIME	更新时间	datetime	无	否	Not null
UPDATED_USER	更新用户	char	10	否	Not null

分类表只有两个属性，编号和分类名，具体如表 4 所示：

表 4 分类表 category

字段	字段名	类型	宽度	是否主键	是否为空
ID	分类编号	int	11	是	Not null
NAME	分类名称	char	20	否	Not null

生活照片表为博主上传的生活照，包含了照片的路径、大小和上传时间，具体如表 5：

表 5 生活照片表 life\_image

字段	字段名	类型	宽度	是否主键	是否为空
ID	编号	char	40	是	Not null
URL	路径	char	255	否	Not null
SIZE	大小	decimal	10	否	Not null
TIME	上传时间	datetime	1500	否	Not null

推荐文章表保存着博客首页推荐文章的信息，包括编号、路径、标题和推荐度，具体如表 6 所示：

表 6 文章推荐表 recommend

字段	字段名	类型	宽度	是否主键	是否为空
ID	编号	char	40	是	Not null
URL	路径	char	255	否	Not null
TITLE	文章标题	char	40	否	Not null
SERIAL	推荐度	int	2	否	Not null

本系统的所有数据库表没有采用外键的策略。理论上某些表存在关系，又一对多或者一对一的关系，但是依然没有采用外键（Roško Z, 2008）。原因是采用外键对于大量数据的查找操作会降低效率，并且数据的安全性交由代码去把控，从而保证数据的正确性（Jeffrey A. Hoffer, 2013）。

## 4 系统实现

### 4.1 系统开发环境

本系统开发环境如下：

- (1) 工作环境：Windows 8;
- (2) 开发工具：IDEA;
- (3) Java 版本：JDK1.8;
- (4) 数据库：MySQL5.5;
- (5) 其他组件：Navicat for MySQL、FTPServer、Nginx

### 4.2 关键技术的实现

#### 4.2.1 数据源的配置和 ORM 的实现

在 Spring Boot 中配置数据源无需配置 XML 文件，在 application.yml 文件中配置相关信息即可，application.yml 文件的配置更加的直观，而且易维护。内容如下：

```
spring:
  datasource:
    driverClassName: com.mysql.jdbc.Driver
    url: jdbc:mysql://127.0.0.1:3306/blogs
    username: root
    password: Dengjinhui123
    initialSize: 5
    minIdle: 5
    maxActive: 20
    maxWait: 60000
    timeBetweenEvictionRunsMillis: 60000
    minEvictableIdleTimeMillis: 300000
    validationQuery: select 1 from dual
    testWhileIdle: true
    testOnBorrow: false
    testOnReturn: false
    poolPreparedStatements: true
    maxPoolPreparedStatementPerConnectionSize: 20
    filters: stat,wall,log4j
    connectionProperties: druid.stat.mergeSql=true;druid.stat.slowSqlmillis=5000
```

MyBatis 的 ORM 实现非常简单，并且 Github 上有插件可以根据数据库表自动生成 Mapper 层和对应的 XML 文件，自动生成的 Mapper 包含最简单的增删查改操作，无需自己手动编写，如果想增加 DAO 层的接口只需生成一个新的 MapperExt 继承 Mapper，再手动生成对应的 XML 文件编写 SQL 语句即可。以用户模块举例，插件自动生成了基本

的 Mapper 类和 XML 的文件，UserMapper 类的内容如下：

```
public interface UserMapper {  
    int deleteByPrimaryKey(String id);  
    int insert(User record);  
    int insertSelective(User record);  
    User selectByPrimaryKey(String id);  
    int updateByPrimaryKeySelective(User record);  
    int updateByPrimaryKey(User record);  
}
```

对应的 UserMapper.xml 内容如下：

```
<?xml version="1.0" encoding="UTF-8" ?>  
<!DOCTYPE mapper PUBLIC "-//mybatis.org/DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd" >  
<mapper namespace="com.deng.customer.mapper.UserMapper" >  
    <resultMap id="BaseResultMap" type="com.deng.customer.entity.User" >  
        <id column="ID" property="id" jdbcType="CHAR" />  
        <result column="USERNAME" property="username" jdbcType="CHAR" />  
        <result column="PASSWORD" property="password" jdbcType="CHAR" />  
        <result column="PROFESSION" property="profession" jdbcType="CHAR" />  
        <result column="PHONE" property="phone" jdbcType="CHAR" />  
        <result column="EMAIL" property="email" jdbcType="CHAR" />  
        <result column="IS_ADMIN" property="isAdmin" jdbcType="INTEGER" />  
    </resultMap>  
    <sql id="Base_Column_List" >  
        ID, USERNAME, PASSWORD, PROFESSION, PHONE, EMAIL, IS_ADMIN  
    </sql>  
    <select id="selectByPrimaryKey" resultMap="BaseResultMap" parameterType="java.lang.String" >  
        select  
        <include refid="Base_Column_List" />  
        from user  
        where ID = #{id,jdbcType=CHAR}  
    </select>  
    <delete id="deleteByPrimaryKey" parameterType="java.lang.String" >  
        delete from user  
        where ID = #{id,jdbcType=CHAR}  
    </delete>  
    <insert id="insert" parameterType="com.deng.customer.entity.User" >  
  
        insert into user (ID, USERNAME, PASSWORD,  
            PROFESSION, PHONE, EMAIL, IS_ADMIN)  
        values (#{id,jdbcType=CHAR}, #{username,jdbcType=CHAR}, #{password,jdbcType=CHAR},  
            #{profession,jdbcType=CHAR},          #{phone,jdbcType=CHAR},          #{email,jdbcType=CHAR},  
            #{isAdmin,jdbcType=INTEGER})  
    </insert>
```



```

<insert id="insertSelective" parameterType="com.deng.customer.entity.User" >
insert into user
<trim prefix="(" suffix=")" suffixOverrides="," >
    <if test="id != null" >
        ID,
    </if>
    <if test="username != null" >
        USERNAME,
    </if>
    <if test="password != null" >
        PASSWORD,
    </if>
    <if test="profession != null" >
        PROFESSION,
    </if>
    <if test="phone != null" >
        PHONE,
    </if>
    <if test="email != null" >
        EMAIL,
    </if>
    <if test="isAdmin != null" >
        IS_ADMIN,
    </if>
</trim>
<trim prefix="values (" suffix=")" suffixOverrides="," >
    <if test="id != null" >
        #{id,jdbcType=CHAR},
    </if>
    <if test="username != null" >
        #{username,jdbcType=CHAR},
    </if>
    <if test="password != null" >
        #{password,jdbcType=CHAR},
    </if>
    <if test="profession != null" >
        #{profession,jdbcType=CHAR},
    </if>
    <if test="phone != null" >
        #{phone,jdbcType=CHAR},
    </if>
    <if test="email != null" >
        #{email,jdbcType=CHAR},
    </if>

```

```

    <if test="isAdmin != null" >
        #{isAdmin,jdbcType=INTEGER},
    </if>
</trim>
</insert>
<update id="updateByPrimaryKeySelective" parameterType="com.deng.customer.entity.User" >
    update user
    <set >
        <if test="username != null" >
            USERNAME = #{username,jdbcType=CHAR},
        </if>
        <if test="password != null" >
            PASSWORD = #{password,jdbcType=CHAR},
        </if>
        <if test="profession != null" >
            PROFESSION = #{profession,jdbcType=CHAR},
        </if>
        <if test="phone != null" >
            PHONE = #{phone,jdbcType=CHAR},
        </if>
        <if test="email != null" >
            EMAIL = #{email,jdbcType=CHAR},
        </if>
        <if test="isAdmin != null" >
            IS_ADMIN = #{isAdmin,jdbcType=INTEGER},
        </if>
    </set>
    where ID = #{id,jdbcType=CHAR}
</update>
<update id="updateByPrimaryKey" parameterType="com.deng.customer.entity.User" >
    update user
    set USERNAME = #{username,jdbcType=CHAR},
        PASSWORD = #{password,jdbcType=CHAR},
        PROFESSION = #{profession,jdbcType=CHAR},
        PHONE = #{phone,jdbcType=CHAR},
        EMAIL = #{email,jdbcType=CHAR},

        IS_ADMIN = #{isAdmin,jdbcType=INTEGER}
    where ID = #{id,jdbcType=CHAR}
</update></mapper>

```

对应的实体类 User.java 内容如下：

```

import lombok.Data;

@Data
public class User {

```

```
private String id;
private String username;
private String password;
private String profession;
private String phone;
private String email;
private Integer isAdmin;
```

通过上述的 ORM 映射，就能把实体类和数据库中的表数据对应起来，要操作数据库的数据操作对象即可，此为面向对象编程的精髓。

#### 4.2.2 Spring Boot 框架搭建

Spring Boot 项目可通过创建一个 Maven 来搭建，POM 文件需要依赖 Spring Boot 的相关 jar 包，POM 文件中核心内容如下：

```
<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>1.5.9.RELEASE</version>
  <relativePath/> <!-- lookup parent from repository -->
</parent>
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-jdbc</artifactId>
  </dependency>
  <dependency>
    <groupId>org.mybatis.spring.boot</groupId>
    <artifactId>mybatis-spring-boot-starter</artifactId>
    <version>1.2.2</version>
  </dependency>
</dependencies>
```

由于是 Web 项目，而且 Spring Boot 内嵌了 Tomcat，只需要在 .yml 文件中设置端口：

```
server:
  port: 9092
```

运行 application 类的 main 方法，该 Web 项目就被运行起来，就像运行一个普通 java 类一样简单，这就是 Spring Boot 的优点之一。

#### 4.2.3 Spring Cloud 微服务搭建

Spring Cloud 微服务框架的搭建是本系统的核心，Spring Cloud 基于 Spring Boot，要创建一个服务注册与发现中心首先创建一个 Spring boot 项目，POM 文件中引入 Eureka 的依赖，依赖如下：

```
<dependency>
  <groupId>org.springframework.cloud</groupId>
  <artifactId>spring-cloud-starter-eureka-server</artifactId>
</dependency>
```

在该项目的.yml 文件中配置 Eureka 的信息，内容如下：

```
eureka:
  instance:
    hostname: localhost
  client:
    registerWithEureka: false
    fetchRegistry: false
  serviceUrl:
    defaultZone: http://${eureka.instance.hostname}:${server.port}/eureka/
```

在该项目的入口类添加注解@EnableEurekaServer，启动入口类的 main 方法，项目启动成功表示在 9090 端口创建了一个服务注册与发现中心，访问 <http://localhost:9090/>，如图所示 7：

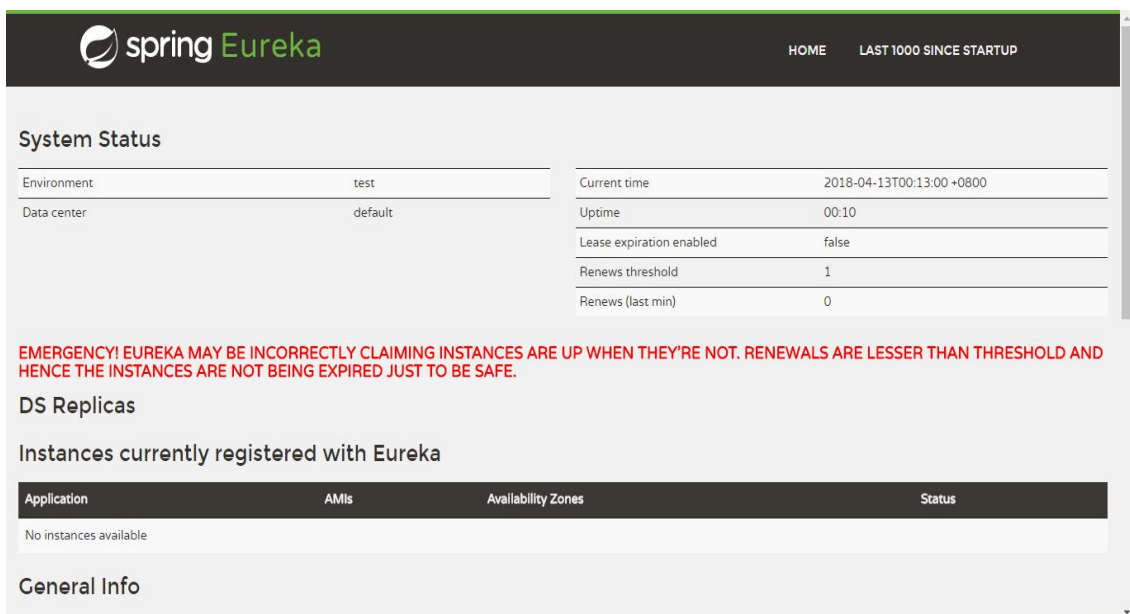


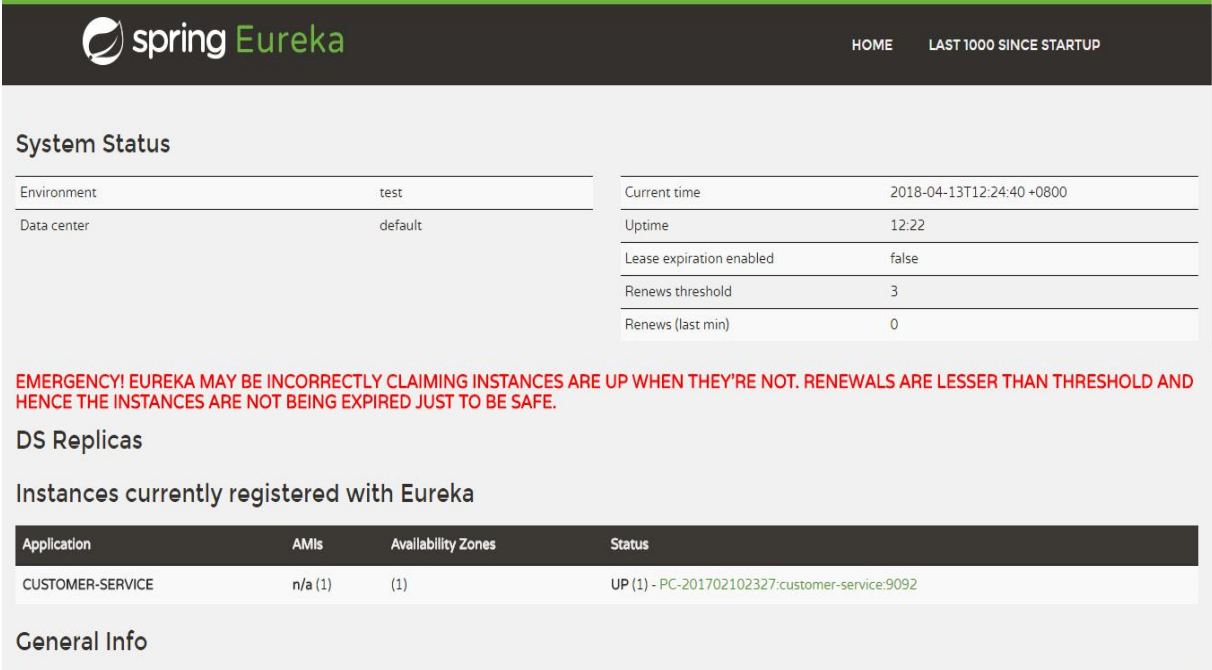
图 7 服务注册中心 Eureka

图 7 表示 Eureka 创建成功，Application 列表为空，因为还没有服务向其注册。

微服务需要被创建并且注册到 Eureka 上才可以正常使用，服务之间的发现由 Eureka 支持，Eureka 组件将维护一份列表，这份列表保存在每个服务的名字和 URL，下面举个例子创建一个服务并且注册到 Eureka。首先创建一个 Spring Boot 项目，POM 文件引入上述 Eureka 的依赖，然后入口类添加注释@EnableEurekaClient，application.yml 文件配置微服务的相关信息，内容如下：

```
eureka:
  client:
    serviceUrl:
      defaultZone: http://localhost:9090/eureka/
server:
  port: 9092
spring:
  application:
    name: customer-service
```

表明服务的名字为 customer-service，向端口为 9090 的 Eureka 注册服务。启动入口类的 main 方法，访问 http://localhost:9090/，如图 8 所示：



The screenshot displays the Spring Eureka web interface. At the top, there's a navigation bar with 'HOME' and 'LAST 1000 SINCE STARTUP'. The main content area is divided into several sections:

- System Status:** A table showing environment (test) and data center (default).
- DS Replicas:** A table showing current time (2018-04-13T12:24:40 +0800), uptime (12:22), lease expiration enabled (false), renewals threshold (3), and renewals (last min) (0).
- EMERGENCY!** A red warning message: "EUREKA MAY BE INCORRECTLY CLAIMING INSTANCES ARE UP WHEN THEY'RE NOT. RENEWALS ARE LESSER THAN THRESHOLD AND HENCE THE INSTANCES ARE NOT BEING EXPIRED JUST TO BE SAFE."
- Instances currently registered with Eureka:** A table with columns: Application, AMIs, Availability Zones, and Status. It shows one instance: CUSTOMER-SERVICE, with status UP (1) - PC-201702102327:customer-service:9092.
- General Info:** A section at the bottom.

图 8 服务注册中心 Eureka

可见 customer-service 已经注册到 Eureka 了，其他模块操作类似。

#### 4.2.4 Spring Session 框架配置

session 是由 Servlet 容器创建和维护的,不同的 Tomcat 容器生成的 session 信息不同,在分布式系统中请求有可能会被指派到不同的服务器处理,因此 session 信息不同会造成用户信息不统一的问题,因此分布式系统需要解决的一个核心难题是 session 信息的统一。本系统中使用 Spring Session+Redis 框架完成微服务系统的 session 统一。首先项目的 POM 文件引入 Spring Session 和 Redis 的依赖,内容如下:

```
<!-- https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-redis -->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-redis</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.session</groupId>
    <artifactId>spring-session-data-redis</artifactId>
</dependency>
```

将下载好的 Redis 服务器安装并且运行,并且创建配置类初始化 Session 的配置,内容如下:

```
import org.springframework.session.web.context.AbstractHttpSessionApplicationInitializer;
//初始化 Session 配置
public class SessionInitializer extends AbstractHttpSessionApplicationInitializer {
    public SessionInitializer() {
        super(SessionConfig.class);
    }
}
```

然后配置 Jedis 工厂类,内容如下:

```
@Configuration
@EnableRedisHttpSession(maxInactiveIntervalInSeconds= 1800)
public class SessionConfig {
    @Bean
    public JedisConnectionFactory connectionFactory() {
        JedisConnectionFactory connection = new JedisConnectionFactory();
        return connection;
    }
}
```

正常运行项目, Spring Session 框架就会工作,自动替换掉 HttpSession 并且将 Session 信息保存到配置好的 Redis 服务器上,前提是 Redis 服务器要正常运行,否则项目将报错。

#### 4.2.5 Swagger UI 搭建

Spring Boot 集成 Swagger UI 框架,使得接口文档的生成更加简便,不需要自己编写 API 文档,而且 Swagger 包含着测试接口的功能。要使用 Swagger UI 框架,首先 POM 文

件引入 Swagger UI 的依赖，内容如下：

```
<!-- https://mvnrepository.com/artifact/io.springfox/springfox-swagger2 -->
<dependency>
    <groupId>io.springfox</groupId>
    <artifactId>springfox-swagger2</artifactId>
    <version>2.7.0</version>
</dependency>
<!-- https://mvnrepository.com/artifact/io.springfox/springfox-swagger-ui -->
<dependency>
    <groupId>io.springfox</groupId>
    <artifactId>springfox-swagger-ui</artifactId>
    <version>2.7.0</version>
</dependency>
```

项目的入口类配置 Swagger 的 bean 并且加上注释，内容如下：

```
@Bean
public Docket createRestApi() {
    return new Docket(DocumentationType.SWAGGER_2)
        .apiInfo(apiInfo())
        .select()
        .apis(RequestHandlerSelectors.basePackage("com.deng.customer.web"))
        .paths(PathSelectors.any())
        .build();
}

private ApiInfo apiInfo() {
    return new ApiInfoBuilder()
        .title("Customer_API")
        .description("")
        .termsOfServiceUrl("")
        .contact("Deng")
        .version("1.0")
        .build();
}
```

搭建完成之后运行项目，访问 <http://localhost:9092/swagger-ui.html#/>，就可以看到该模块的 Controller 的 API 文档，如图 9 所示：



图 9 Swagger UI 首页

由于项目的前后端分离开发，使用 Json 数据交互，因此，一款优秀的 API 文档生成框架和测试工具显得十分重要，本系统 Swagger 就充当了这两个角色。

#### 4.2.6 AOP 接口鉴权实现

Spring 的特性之一是 AOP，即面向切面编程，本系统使用 AOP 技术完成接口鉴权的功能。某些接口如评论接口，需要先验证该用户是否在登录的状态。它实现方式基于 Spring Session，Spring Session 会在浏览器第一次访问服务器的时候创建一个 session，并且保存在 Redis 服务器，最后将该 session 的 ID 返回给客户端，下一次浏览器访问将带着这个 sessionId。当用户登录之后，会随机生成一个 token，并将 userId 连同 token 保存在 session 中，再保存到 Redis 服务器。等下一次该用户访问部分需要验证权限的接口时带上 userId，服务器会从判断 Redis 中是否存在该 userId 的 token，若有，证明通过验证，否则无权访问。

下面演示留言的接口测试，在没有登录的情况下，输入要留言的文章 ID 为 123，测试结果如图 10 所示：





图 10 留言接口测试（未登录）

结果表明 Spring Session 框架起作用，这时任意用户进行登录，如图 11 所示：



图 11 登录成功图

用户登录成功后，返回该用户的 ID，这时再测试留言的接口，userId 作为参数之一，结果如图 12 所示：



图 12 留言接口测试（已登录）

此时打开 Redis 客户端输入 `keys *` 命令可查看 Redis 所有数据，如图所示：

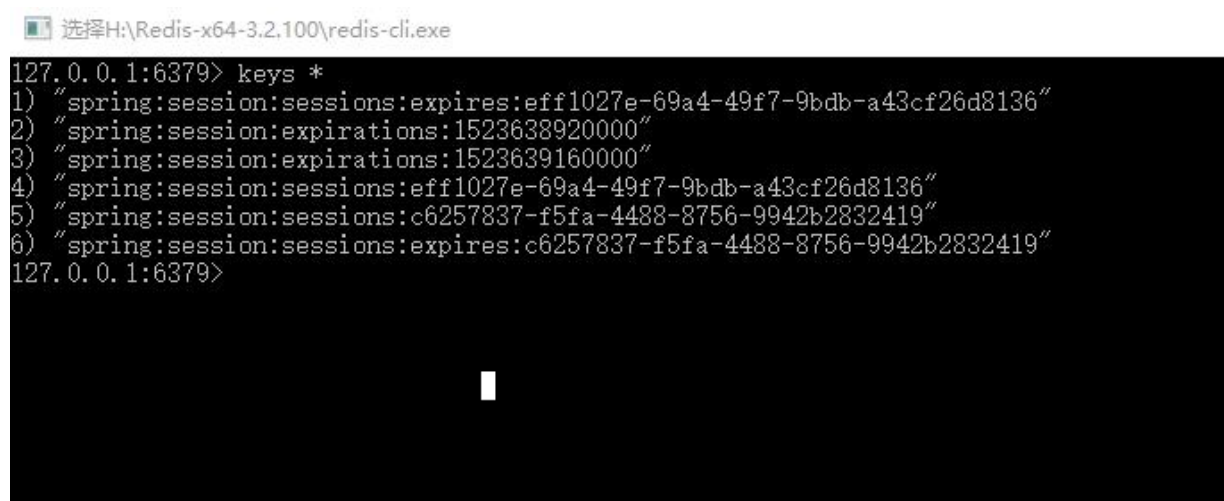


图 13 Redis 数据库

如图所示，Redis 中存在两个 Session 信息，代表有两个不同的用户从不同的浏览器进行访问。

## 5 运行效果

### 5.1 博客首页



图 14 博客首页 1



图 15 博客首页 2

博客首页如图 14、15 所示，首页的内容主要是博客文章，并且这些文章根据创建日期降序排序。首页还包含着博客推荐的文章，这些文章由管理员管理，链接着其它网站的著名文章。右上角包含登录和注册的按钮，点击按钮将会对应登录和注册的操作。首页有不同的分类，不同的分类有不同的文章，如技术框架：



图 16 技术框架文章

5.2 文章详情

当点击文章的标题时，将显示文章的详细内容，如图 17、18 所示：

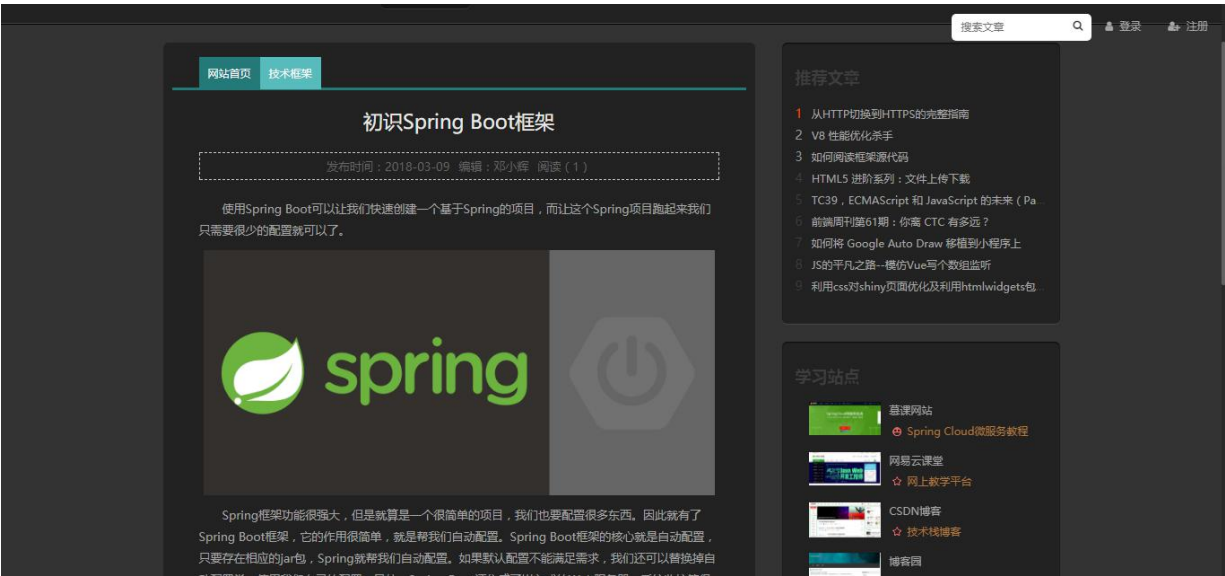


图 17 文章详情 1

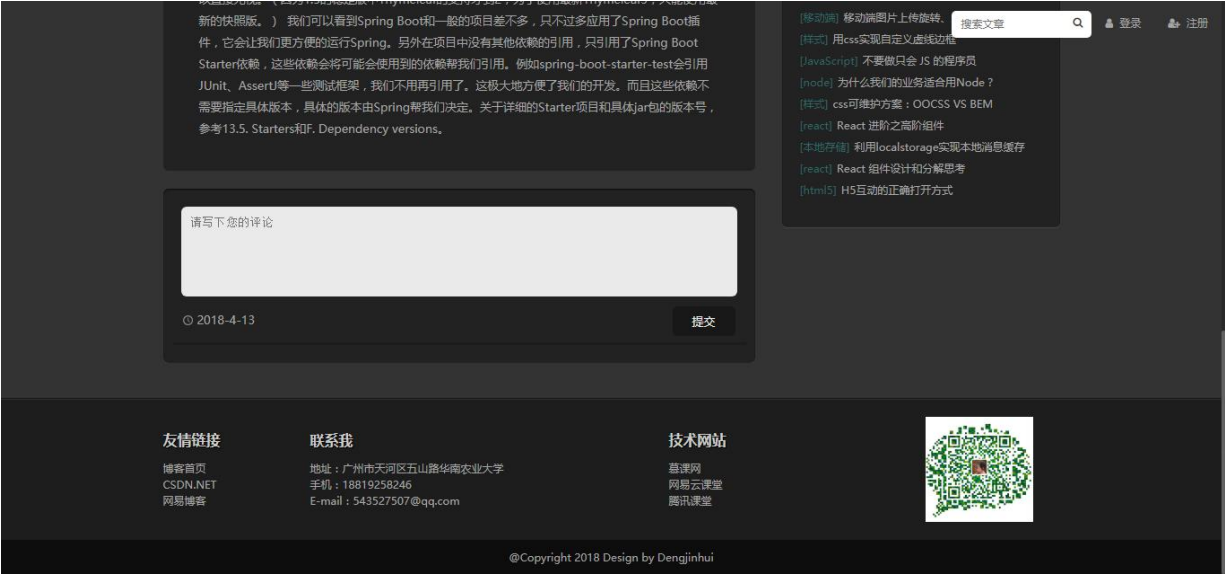


图 18 文章详情 2

文章详情包含文章的基本信息,如标题、简介、图片、发布时间和阅读量等等。用户登录如图 19 所示:

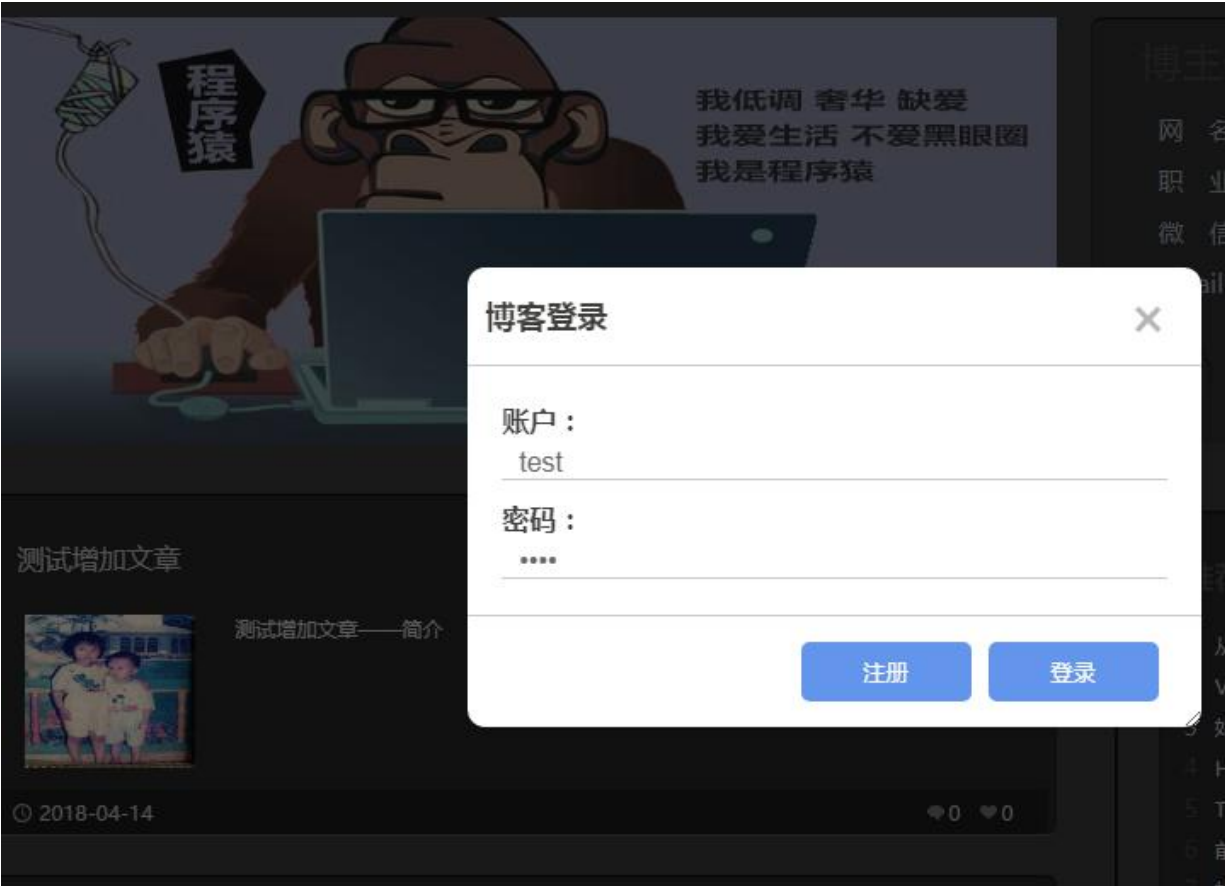


图 19 登录界面

当用户登录后，可以对文章发表留言评论，评论完成后评论区如图 20 所示：

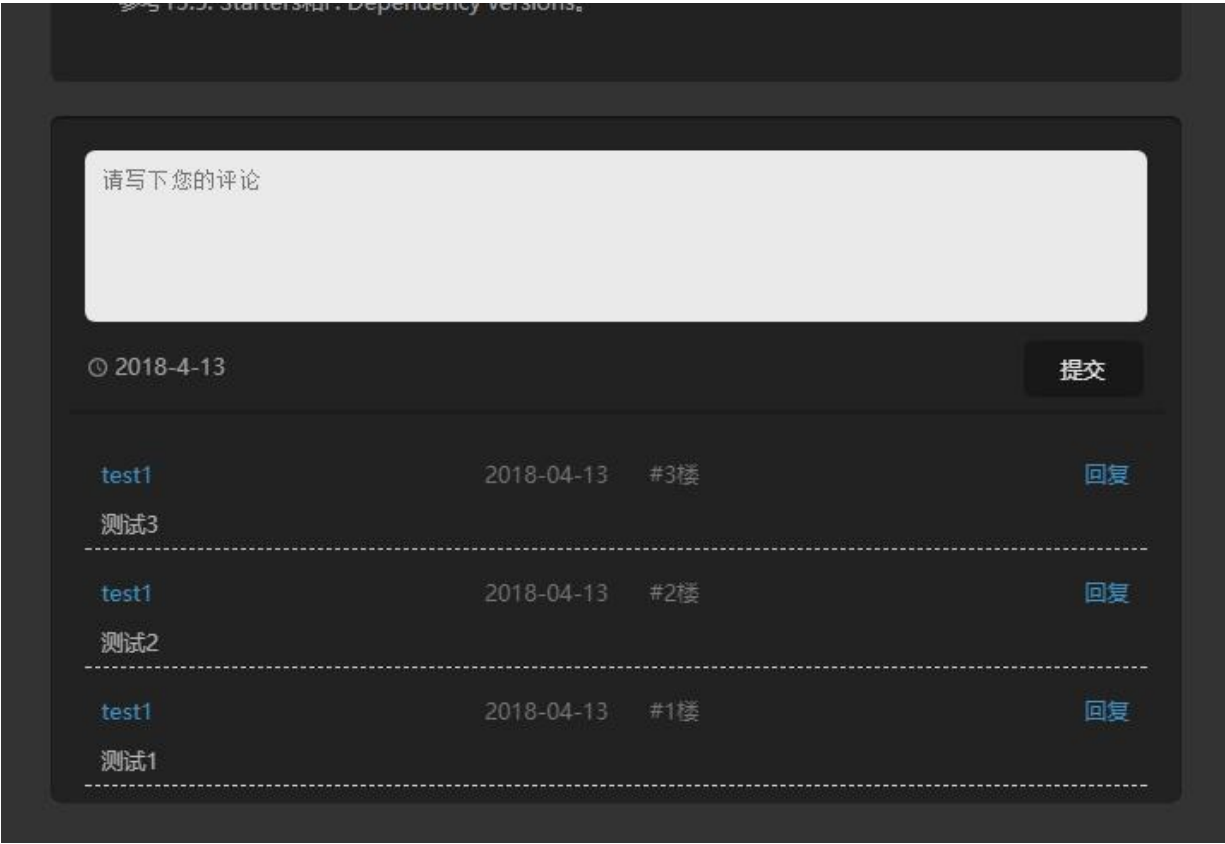


图 20 评论功能

另一个用户可以对某条评论进行回复，点击回复按钮时，评论框会自动截取被回复者的信息，如图 21 所示：

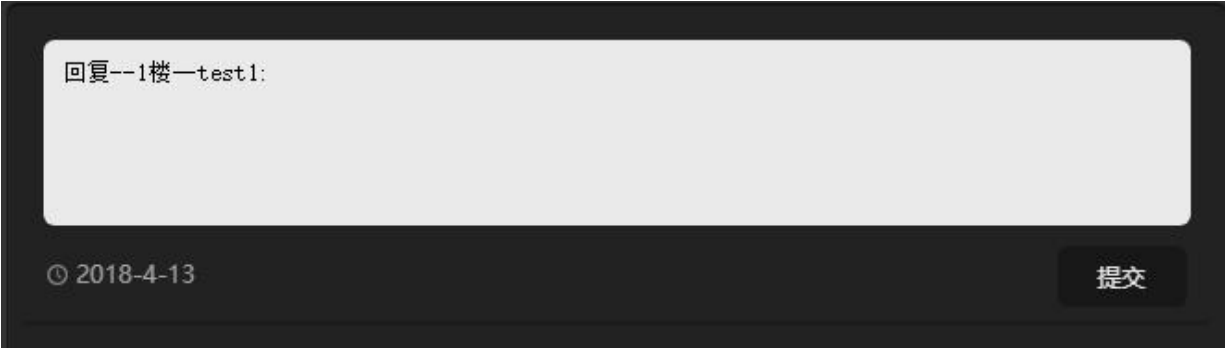


图 21 回复评论演示

输入完回复的信息之后，点击提交，回复成功后评论区自动刷新，如图 22 所示：

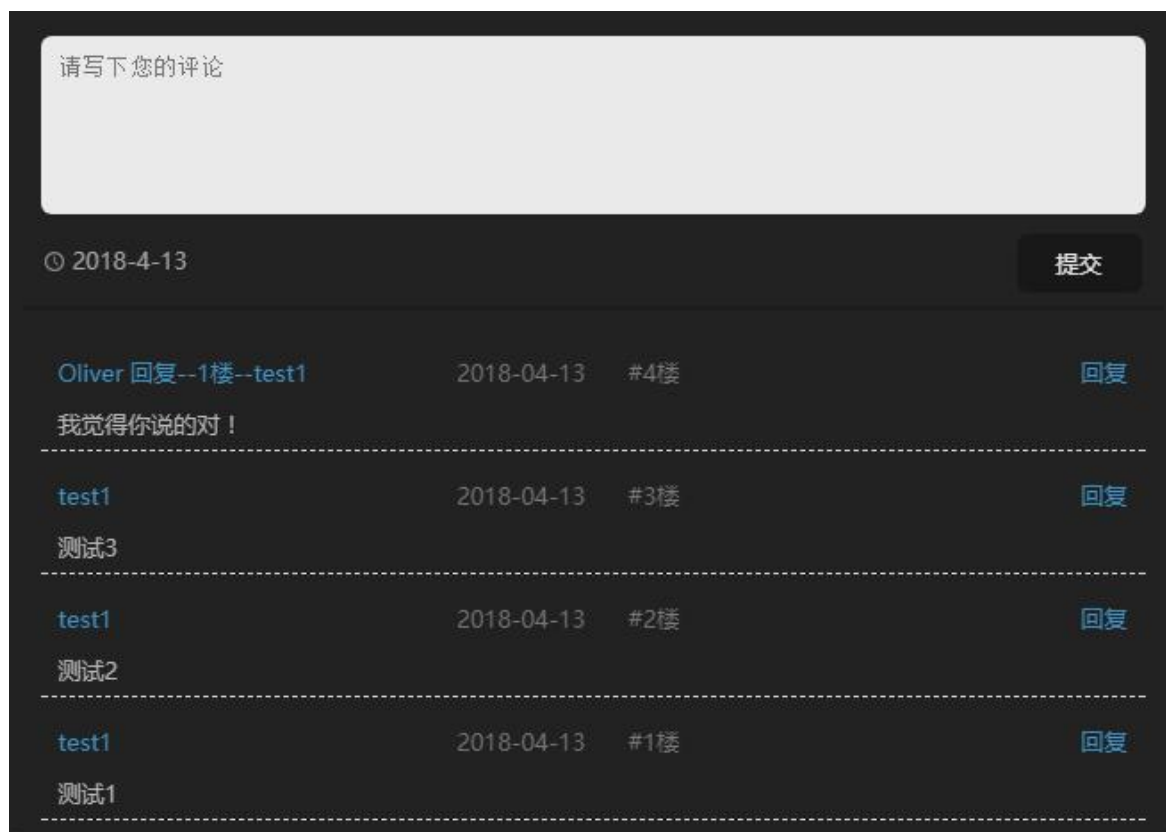


图 22 回复评论成功

### 5.3 系统管理员

当用户登录的用户名和密码是管理员的用户名和密码时，系统跳到后台管理页面，如图 23 所示：

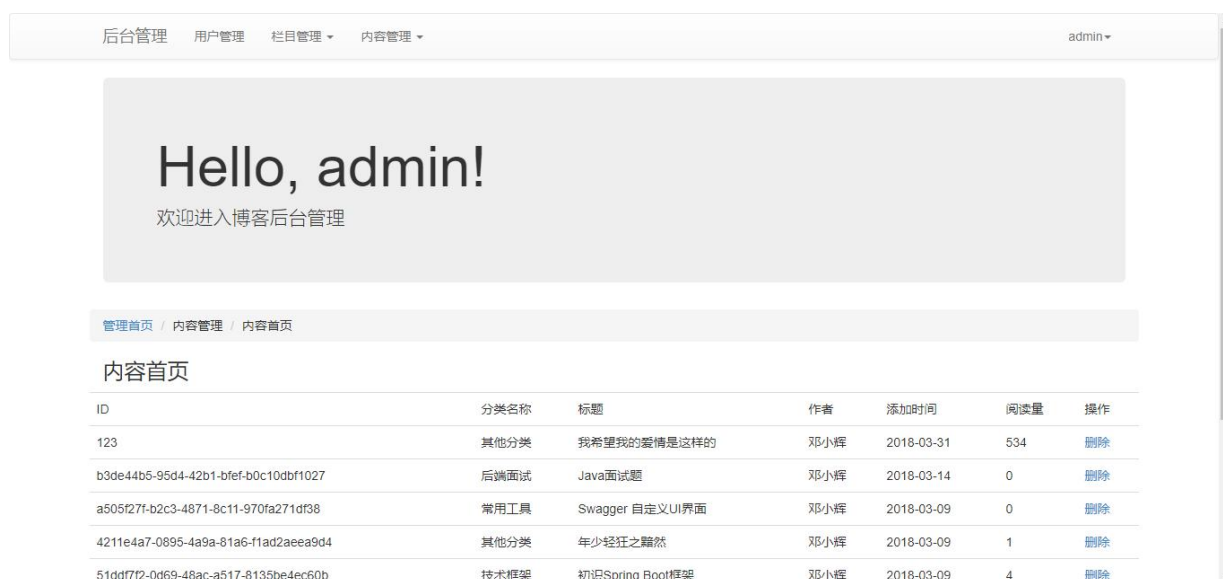


图 23 后台首页



管理员可以管理文章模块、分类模块和用户模块，拥有对上述模块内容的增删查改操作，例如增加文章如图 24 所示：

后台管理

用户管理

栏目管理

内容管理

管理首页

内容管理

内容添加

内容添加

分类：

全栈开发

标题：

测试增加文章

选择文件

微信图片\_20180...214901.jpg

简介：

测试增加文章——简介

内容：

测试增加文章——内容

图 24 增加文章

选择分类为全栈开发，分别写上标题、图片、简介和 content 等信息即可提交，提交成功后回到首页，文章已被添加，如图 25 所示：



图 25 新增文章成功



## 6 总结与展望

### 6.1 总结

本系统基于 MVC 思想，后端采用了 Spring Boot+MyBatis 框架开发服务模块，Spring Cloud 框架开发微服务系统，前端 js 框架用到 Bootstrap、jQuery，是一个完整的个人博客网站。系统的业务功能完整，前端的页面较友好，后端的开发技术较主流。

现将本次毕业设计中的收获与结论总结一下：

（1）查阅 Spring Boot 框架的相关知识，了解到了 Spring Boot 的好处和它的高效性，跟着网上的教程最终能做到快速搭建 Spring Boot 项目。

（2）在系统开发前在草稿纸上简述业务功能和系统的流程，经过多次修改后用 Visio 严谨地画出流程图、模块图和系统框架图。

（3）搭建 Spring Cloud 微服务的过程中翻阅许多资料，因为第一次搭建微服务，每一步都需要非常仔细和严谨。

（4）借助优秀的软件开发系统，如 Navicat for MySQL 管理数据库、Visio 画流程图、FTP Server 处理文件的上传和下载等等。

（5）针对 session 信息统一的解决方案采用了 Spring Session 框架替换 HttpSession 的方法，框架自定义过滤器，重写了 getSession 方法，并且将 session 信息保存到 Redis。

### 6.2 展望

本个人博客系统完成了大部分的设计与实现，但仍然有一些问题有待进一步的完善：

（1）本系统的业务符合一个合格的个人博客系统的要求，但是缺乏创新性的功能。

（2）本系统没有进行性能检测、调优工作，只完成了功能上的实现，在性能方面不足以成为一个能够上线的网站。

（3）本系统没有考虑有关网络攻击的问题，可能会出现黑客攻击、安全漏洞等问题。

（4）由于本人前端知识欠缺，前端页面美观性仍然不够。

## 参 考 文 献

- 毕建信. 基于 MVC 设计模式 WEB 应用研究与实现[D]. 武汉: 武汉理工大学, 2006.
- 陈雄华. Spring 企业级应用开发详解[M]. 北京: 电子工业出版社, 2009:66-88.
- 贺松平. 基于 MVC 模式的 B/S 架构的研究与应用[D]. 武汉: 华中科技大学, 2009.
- 贾文潇, 邓俊杰. 基于 Java 的 Web 开发技术浅析[J]. 电子测试, 2016, (8): 65-86.
- 刘京华. Java Web 整合开发王者归来[M]. 北京: 清华大学出版社, 2010:56-78.
- 孙卫琴, 李洪成. Tomcat 与 Java Web 开发技术详解[M]. 北京:电子工业出版社, 2003:160-205.
- 谢世波. J2EE 数据持久层的解决方案[J]. 计算机工程, 2003, 6(22):93-95.
- 杨静. 基于 JAVA WEB 中 MVC 模式的研究与应用[J]. 电脑知识与技术, 2014, 9(28):68-71.
- 俞琰, 郑阿奇.J2EE 应用实践教程[M]. 北京: 电子工业出版, 2009:100-112.
- 张黎明, 龚琪琳. 基于 MVC 模式的 Java Web 应用设计[J]. 计算机与现代化, 2007, (2): 22-24.
- Azat Mardan. Redis and Authentication Patterns[M]. New York: Apress, 2014.
- Bruce Eckel. Thinking in Java[M]. Upper Saddle River, New Jersey, USA:Prentice Hall, 2006:45-97.
- Craig Walls, Ryan BreidenBach. Spring In Action [M]. Manning Publications, 2006.
- Design[M], Third Edition.Pearson Education, 2013: 45-50.
- Jeffrey A.Hoffer, University of Dayton, Joey F.George.Modern Systems Analysis and Roško Z, Konecki M. Dynamic Data Access Object Design Pattern (CECIIS 2008)[J]. Central European Conference on Information & Intelligent Systems, 2008.47-52.
- Peter Ljungstrand. Awareness of presence, instant messaging and WebWho[J]. ACM SIGGROUP Bulletin, 2000, 21(3): 21-27.

## 致 谢

经过几个月的努力，本人完成该个人博客系统的开发，我能够成功一部分源于身边的亲朋好友，在此我要对他们真挚地致谢。

首先我要感谢指导老师刘汉兴老师，老师在我选题之后提出该题目普通可见，需要做到与众不同，或者用到与众不同的技术才能拿到高分，否则就是一份普通的毕业设计。在老师的建议和鼓励之下，我翻阅大量资料，浏览主流的技术博客，最终决定使用现今主流的框架和技术。在我遇到困难时，老师总会给我提供帮助，他会用实际行动引导我并且鼓励我，在此非常感谢刘汉兴老师给予帮助和支持。

感谢亲人一直以来对支持。从小到大亲人总会满足需求，在学业上对我更是给予无限的支持，任何关于学业的花费都会二话不说地提供给我。当我处于低落期时亲人会带我去放松，教我人生道理，引导我慢慢回到正轨。在此真挚地感谢亲人。

感谢给予我帮助的小伙伴们，感谢王伊莹同学的指导，在我迷茫的时候鼓励我、帮助我。感谢彭梓晖同学的帮助，在我有疑惑的时候提点我，还向我推荐好的书籍。

感谢所有帮助过我的人，你们的帮助让我离成功又近了一步。

学号		姓名		专业		
毕业论文题目						
指导教师评语						
成绩（百分制）：_____ 指导教师签名：_____ 年 月 日						
评 阅 人 评 语 及 成 绩 评 定	成绩 评定 标准	评分项目			分值	得分
		选题 质量 20%	1	专业培养目标	5	
			2	课题难易度与工作量	10	
			3	理论意义或生产实践意义	5	
		能力 水平 40%	4	查阅文献资料与综合运用知识能力	10	
			5	研究方案的设计能力	10	
			6	研究方法和手段的运用能力	10	
			7	外文应用能力	10	
		成果 质量 40%	8	写作水平与写作规范	20	
	9		研究结果的理论或实际应用价值	20		
	评阅人评语					
	成绩（百分制）：_____ 评阅人签名：_____ 年 月 日					

续上表:

	评价项目	具体要求 (A 级标准)	最高分	评分						
				A	B	C	D	E		
答辩小组评语及成绩评定	论文质量	论文结构严谨, 逻辑性强; 有一定的学术价值或实用价值; 文字表达准确流畅; 论文格式规范; 图表(或图纸) 规范、符合要求。	60	55-60	49-54	43-48	37-42	≤36		
	论文报告、讲解	思路清晰; 概念清楚, 重点(创新点) 突出; 语言表达准确; 报告时间、节奏掌握好。	20	19-20	17-18	15-16	13-14	≤12		
	答辩情况	答辩态度认真, 能准确回答问题	20	19-20	17-18	15-16	13-14	≤12		
成绩总评	答辩小组评语  是否同意通过论文答辩 (打√)  1. 同意  2. 不同意  成绩 (百分制): _____ 答辩小组成员 (签名): _____  <div style="text-align: right;">年 月 日</div>									
	论文总评分数: _____  教学院长签名: _____ 学院盖章: _____  <div style="text-align: right;">年 月 日</div>									