基于Spring Boot框架的漫画网站开发

摘要：现如今随着互联网的发展，目前国家“互联网+”战略的部署，全球第五代移动通信技术开始全面向大众普及。人们通过互联网获取信息更加的多元化。而其中“二次元”作为一种新型的网络文化在网络当中传播开来，影响了国内的大部分年轻群体，深受年轻人的喜爱。因此，本次课题为发扬互联网共享精神，通过以Spring Boot为基础的后端开发框架和Boot Strap为基础的前端框架相结合，设计了一个为广大青少年二次元文化爱好者以及创作者服务的漫画在线阅读分享平台。而在开发过程当中，由于整个系统当中最为重要的载体漫画是以图片的方式呈现的，考虑到图片的质量以及容量大小，因此在这基础上外接了图床接口，通过以图床的方式上传与加载以此来改善系统的流畅度，增强用户的使用体验。

关键词： 漫画；Spring Boot；Boot Strap；图床；网站

Comic website development based on Spring Boot framework

Abstract: Nowadays, with the development of the Internet and the deployment of the national "Internet +" strategy, the world's fifth-generation mobile communication technology has begun to spread to the general public. People get more diverse information through the Internet. Among them, the "two-dimensional" as a new type of network culture spread through the network, affecting most of the young groups in China, and is loved by young people. Therefore, in order to carry forward the spirit of Internet sharing, this topic designed a comic that serves the majority of young secondary culture fans and creators by combining the back-end development framework based on Spring Boot and the front-end framework based on Boot Strap. Online reading and sharing platform. In the development process, because the most important carrier comics in the entire system are presented in the form of pictures, considering the quality and capacity of the pictures, on this basis, an external bed interface is connected and uploaded by the bed And loading to improve the fluency of the system and enhance the user experience.

Keywords: Comic; Spring Boot; Boot Strap; Figure bed; website

目录

摘要 i

Abstract i

[目录 iii](#_Toc31214)

[1 绪论 1](#_Toc8368)

[1.1 课题的目的和意义 1](#_Toc16297)

[1.2 国内外研究现状与水平 1](#_Toc23213)

[1.3 发展趋势 2](#_Toc19656)

[2 分析 3](#_Toc27700)

[2.1 系统可行性分析 3](#_Toc12070)

[2.2 系统性能和环境要求 3](#_Toc15197)

[2.3 系统功能分析 3](#_Toc11651)

[2.3.1 系统功能概述 3](#_Toc4324)

[2.3.2 系统详细功能描述 4](#_Toc32440)

[2.3.3 系统基本功能要求 4](#_Toc23812)

[2.3.4 系统功能重点及其难点 5](#_Toc15290)

[2.3.5 系统用户分析 6](#_Toc8186)

[3 系统总体设计 8](#_Toc27265)

[3.1 系统功能模块划分 8](#_Toc31688)

[3.2 数据库设计 10](#_Toc12617)

[4 系统详细设计与编码实现 14](#_Toc24823)

[4.1 程序结构 14](#_Toc31949)

[4.2 系统注册功能的开发 15](#_Toc5725)

[4.2.1 系统注册功能前端代码的设计 15](#_Toc31082)

[4.2.2 系统注册功能后端具体业务实现 16](#_Toc7226)

[4.3 系统登录功能的开发 18](#_Toc32335)

[4.3.1 系统登录功能前端代码的设计 18](#_Toc11568)

[4.3.2 系统登录功能后端具体业务实现 18](#_Toc17182)

[4.4 漫画投稿功能的开发 19](#_Toc29650)

[4.4.1 漫画投稿功能前端代码的设计 19](#_Toc15277)

[4.4.2 漫画投稿功能后端具体业务实现 25](#_Toc485)

[4.5 我的投稿前台管理功能的开发 28](#_Toc4815)

[4.5.1 我的投稿前台管理页面的设计 28](#_Toc16017)

[4.5.2 我的投稿后端具体业务实现 29](#_Toc17472)

[4.6 分类页面功能的开发 30](#_Toc27396)

[4.6.1 分类页面的前台页面的设计 30](#_Toc5939)

[4.6.2 分类页面的后端具体业务实现 31](#_Toc6497)

[4.7 实体漫画浏览功能的开发 32](#_Toc1557)

[4.7.1 实体漫画浏览页面的前台页面的设计 32](#_Toc4079)

[4.7.2 实体漫画浏览页面的后端具体业务实现 34](#_Toc17779)

[4.8 首页功能的开发 34](#_Toc8923)

[4.8.1 首页页面的前台页面的设计 34](#_Toc27447)

[4.8.2 首页页面的后端具体业务实现 35](#_Toc8472)

[5 结论 36](#_Toc141)

[5.1 系统的不足 36](#_Toc5398)

[5.2 系统的完善 37](#_Toc23600)

[5.3 总结与体会 37](#_Toc24688)

[致谢 38](#_Toc20471)

[参考文献 39](#_Toc12733)

# 绪论

旨在介绍漫画网站开发的目的和意义、国内外研究现状与水平以及发展趋势等主要内容。

## 课题的目的和意义

今年来随着科技的日新月异，人们的生活水平也在逐渐提高。人们拥有了更多的获取信息的方式以及获取信息的速度。而其中“二次元”作为一种新型的网络文化在网络当中传播开来，影响了国内的大部分年轻群体，深受年轻人的喜爱。“二次元文化”泛指动画(Anime)、漫画(Comic)、游戏(Game)、轻小说(Novel)内容及相关衍生品。互联网的发展扩大了二次元文化覆盖面和传播广度，更多非核心层次二次元人群有机会接触、了解二次元内容及文化，泛二次元群体不断扩大。而这当中漫画这块还是块未被充分挖掘的沃土。当代年轻人缺少一个漫画资源齐全，使用体验良好，价格亲民的漫画阅读平台。同时国内各漫画创作者们也缺少一个可以供他们分享自己漫画，实现自身梦想的漫画创作平台。因此，本次课题为发扬互联网共享精神，打算设计一个为广大青少年二次元文化爱好者以及创作者服务的漫画在线阅读分享平台。普通用户可以在上边查阅自己喜欢的漫画，对于自己喜欢的漫画可以点赞、收藏、分享给其他人。创作者也可以分享自己所绘制的漫画供别人观赏。

## 国内外研究现状与水平

随着互联网的普及，以及目前国家目前“互联网+”战略的部署，互联网的运用，在国内外的各个行业得到了广泛的运用。漫画作为曾经在纸质媒体上传播的内容，如今也可以通过互联网更快更广的进行传播。通过网络在线观看，也是现在对于纸质媒体开始落寞，提供给漫画一个更加健康的发展方向。根据CNNIC2019发布的报告，我国的网民数量已经增长到了8.54亿，互联网的普及率达到了61.2%，网民使用手机上网的比率更是达到了99.1%。如今的互联网技术可以说是已经十分的成熟了，通过以Spring Boot架构为核心的前后端分离技术可以更好的实现前端和后端的开发工作。Spring作为重量级企业开发框架 Enterprise JavaBean（EJB） 的替代品，Spring为企业级Java开发提供了一种相对简单的方法，通过 依赖注入 和 面向切面编程 ，用简单的 Java对象（Plain Old Java Object，POJO） 实现了EJB的功能。同时市面上也有许多成熟的商业软件和网站像是哔哩哔哩动画，快看漫画，腾讯漫画等等，他们都为我们的开发提供了一个非常好的指导性，可见该程序的开发前景也是非常可观的。

## 发展趋势

在线漫画网站平台有着十分巨大的市场潜力。随着经济的发展，网络科技的进步，人们获取信息的方式也已经从传统的纸质媒体转变为了以互联网为主的新媒体。人们通过互联网能够更加便捷、快速、高效的获取信息内容。这更有利于漫画内容的传播，也为创作者提供了一个更广大的平台去分享自己的作品。另一方面，我国网民的数量已经增长到了8.54亿，其中39岁以下的群体占了总体的70.8%。而其中“二次元文化”是当下深受年轻人所喜爱的流行文化之一，必然有着十分庞大的潜在用户群体。漫画作为二次元文化的其中一部分，目前仍还没有被完全开拓。通过建立在线漫画网站，吸引越来越多的用户观看，这样也能够促进创作者在平台上发布自己的作品以便带来更多用户，实现一套良性的循环。在后期如果市场有需求，也将会慢慢的把平台放眼到更广的市场上去服务更多的人，慢慢的就必定会促使一个认知度高，专业度高，完善度高，服务于更多群体的漫画创作传播平台。

# 分析

## 系统可行性分析

通过对周边学校学生的询问调研发现：如今的大部分学生都有接触过哔哩哔哩动画、Acfun等二次元潮流文化在线娱乐平台，学生们经常会在这些平台上边看自己喜欢的动画、直播、游戏实况等等。但由于这些平台上对于漫画这块还不够重视，许多优质的漫画内容无法在该些平台上找到，他们也希望可以拥有一个更好的在线漫画阅读平台。并且由于漫画与动画的紧密关系，优质的漫画作品将会吸引动画厂商进行动画化，而动画化的作品也会进一步的推动漫画作品的销量。彼此之间产生一种良性的循环，促进双方共同发展，产生更多的效益。所以开发一个基于广大年轻用户为主要目标人群的在线漫画阅读与创作平台是十分必要的。此外系统所用到的两大核心框架Spring Boot和Boot Strap都是目前市面上针对后端和前端两方面的主流框架。是已经经过众多新型互联网企业开发选择下来了的相对成熟稳定的技术。特别是Spring Boot为首的Spring家族框架目前已经是Java后端开发几乎必须要掌握的一门框架了。通过Ioc以及Aop也就是依赖注入以及切面编程两大核心技术点，大大降低了代码之间的耦合程度，提升了代码的执行效率，并在此基础上，通过Spring Boot开箱即用的特性，以及天然内置的Tomcat服务器，减少了从前所需要的编写配置文件的过程，极大的减少了我开发系统的过程，提高了代码开发的编程效率，使得整个系统的开发难度得到进一步的降低。

## 系统性能和环境要求

本系统由于是一个Web应用程序，因此对于电脑的性能需求相对较低。满足如下条件即可。

操作系统：目前主流的图形化操作系统即可

电脑硬件配置：当前主流的电脑配置即可

显示器：分辨率至少在 1024\*768 以上，有条件尽量使用宽屏

网络：通过 互联网 可访问

浏览器：目前主流的以Chrome核心为主的浏览器，如：谷歌，Edge（Chrome版），火狐等

## 系统功能分析

### 系统功能概述

本系统主要分为三大功能：登录功能，投稿功能，漫画浏览功能。分别用于实现用户进入系统，用户上传漫画以及用户浏览漫画。

### 系统详细功能描述

本系统主要涵盖如下各大功能：

#### 登录功能：用户在使用本系统的任意其他功能之前，必须首先进行身份的验证。因此在用户进行其他操作之前必须首先对其是否登录进行校验。登录可通过附属的注册功能以完成相关操作。为保证数据的正确性、安全性，在注册之前必须对用户输入的数据，如：身份证、电话、邮箱等进行合法性校验，并对密码进行MD5加密防止被他人获取盗取后轻易使用。

#### 投稿功能：用户可以在系统平台上共享自己的漫画，可通过投稿上传或者分享自己所拥有的漫画，并可以通过编辑、删除等功能对自己所投稿出去的漫画进行相应的操作去修改其中的信息。

#### 漫画浏览功能：用户可以通过点击对应的漫画对其进行浏览阅读，同时对漫画进行一个历史记录的保存，方便用户在以后可以在历史记录的页面当中选中自己曾经阅读过的书籍进行再阅读。而在这当中如果有用户特别喜欢的漫画，可以对其进行收藏操作，以便用户以后可以更加方便的找到自己喜欢的漫画书。

#### 排行榜功能：系统可以统计每一本漫画的收藏数，并对其进行一个降序排列，方便用户可以更快的找到最受大众喜爱的漫画。

### 系统基本功能要求

本系统各大功能所需实现的基本要求如下

（1）登录功能：

* 注册
* 密码md5加密
* 密码解密
* 身份证校验
* 通过身份证判断生日
* 通过身份证计算年龄
* 通过身份证判断性别
* 手机号校验
* 是否登录检验
* 用户信息修改
* 是否已经注册校验
* 数据录入

1. 投稿功能

* 图片上传图床
* 前端文件上传框架
* 数据前台键值转换
* 漫画的增删改查
* 漫画话数排序
* 自定义话数名称

1. 漫画浏览功能

* 分页功能
* 漫画多条件动态查询
* 漫画名模糊查询
* 前台页面异步回调后台漫画数据
* 排行榜计算

### 系统功能重点及其难点

首先就如上述前文中所提到的，由于本课题的目的是完成一个漫画网站，其最为核心的载体肯定是静态的图片资源，那么如何优化高分辨、大容量的图片资源，对不同格式的图片格式进行统一转换，对千奇百怪的图片尺寸进行统一处理等等这些都是重中之重。因此在本次的课题当中，我所采用了七牛云的图床技术对图片进行统一的管理与处理。七牛云的对象存储功能提供了相对高可用性和高可靠性的存储服务，支持对存储对象的弹性扩容机制，并对可对其进行7×24小时的在线服务，最大化的节省存储的成本。



图 2.1 本系统所用的七牛云对象存储

不仅如此，在七牛云当中提供了格式各样的接口，方便我们对上传好了的图片进行统一化处理。比如在七牛云当中提供了图片样式的接口，方便我们统一对上传了的图片进行缩略裁剪，图片水印，格式输出等个性化处理。保证了系统在我们用户各式各样图片上传后依然得以风格统一。在另一方面，七牛云提供了十分详细且人性化的开发者文档，无论是用Java进行开发，还是C#，PHP等等。基本涵盖了目前市场上任何主流程序开发语言的SDK，提供文件从服务端直接上传七牛的功能，提供对七牛空间中文件进行处理的功能等等方便我们以此来进行开发。并在开发文档当中，例举了大部分常用场景下的样例供我们参考。以此构建我的网络应用程序，可以以非常便捷的方式将图片数据安全地存储到七牛云的平台当中去。且无论是网站程序还是从云端到终端的架构服务或应用，通过七牛云及其SDK，都可以使我的应用程序的终端用户实现高速的上传和下载，同时也能够减轻我的数据库的负荷。



图 2.2 七牛云提供的Java SDK

其次，漫画由于是由一话一话组成的，而每一话又涵盖了多张图片。因此在项目的实际开发过程当中对漫画表进行了拆分。将其拆分成了t\_comic表与t\_comic\_detail表两张表。在t\_comic表中主要用于存储漫画的基础信息，比如作者，出版社等信息，而在t\_comic\_detail表中主要用于存储漫画每一话的信息并设置外键comic\_id关联到主表，至此两张表形成了一个一对多的关系。而在本次的系统当中，由于提供给了用户可以自由上传漫画以及对漫画进行自由编辑的权限。因此当用户对一本漫画的其中一话进行编辑、删除、插入后如何对整体进行排序就成了需要解决的一个点。此外由于存在长篇漫画的形式。故可能会存在一本漫画有近上千的话数，而每一话又有多张图片，如果全部展现，就势必会造成前端页面过于狭长。因此如何设计好前端的页面保证在编辑漫画当中整体的布局人性化，改善用户体验也是在本次课题当中需要解决的一个难点之一。

### 系统用户分析

本系统是主要面向广大15-30周岁的青年人群。该人群基本拥有基础的计算机科学文化素养，因此不必过于担心本系统的操作问题，会给予他们带来困难。并且由于是新时代的受到潮流文化影响的新青年，热爱追捧新兴文化，乐于在互联网上分享自身的经历，拥有充分的互联网共享精神，故统一允许系统内所有注册账号用户拥护投稿权限进行分享创作。但同时考虑到，该类用户往往缺少主观判断，且法律意识较为淡薄。为防止出现侵权行为，黄暴等不健康行为的出现，在注册时强制要求进行实名制登记，如若出现这类情况时，将由管理人员进行封停账号，并将信息上报举报。

# 系统总体设计

## 系统功能模块划分

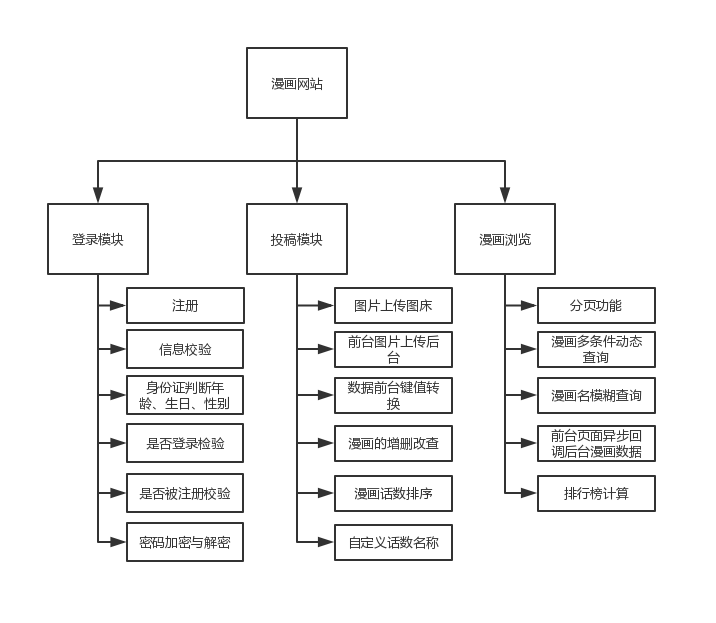


图 3.1 系统功能模块划分

## 数据库设计

在本次系统开发过程当中,我选用了Mysql作为唯一的关系型数据库对本系统的所有数据进行存储操作。Mysql最为当下最为流行的开源数据库之一，有着性能高、成本低、可靠性好等诸多特点，特别适用于对中小型网站数据的存储。且Mysql社区版现在依然是免费提供给大众的，对于目前仍是学生的本人来说是十分友善的。

此外为了对数据高效的进行增删改查等操作，故对数据库进行了如下设计。

#### user:用于存储用户的基础信息

表 3.1 数据库表user

|  |  |  |
| --- | --- | --- |
| **字段名称** | **字段类型** | **备注** |
| Id | Bigint | 用户id |
| Birthday | Datetime | 用户生日 |
| CreateTime | Datetime | 创建时间 |
| Email | Varchar（255） | 邮箱 |
| Gender | Int | 性别 |
| idCard | Varchar（255） | 身份证 |
| isAdm | Int | 是否管理员 |
| isUse | Int | 是否在使用（0：被停封 1：正常） |
| Password | Varchar(255) | 密码 |
| Phone | Varchar(255) | 手机号 |
| realName | Varchar（255） | 真实姓名 |
| userName | Varchar（255） | 用户名（登录用） |

#### t\_comic:用于存储漫画的基础信息

表 3.2 数据库表t\_comic

|  |  |  |
| --- | --- | --- |
| **字段名称** | **字段类型** | **备注** |
| Id | Bigint | 漫画id |
| Address | Varchar（255） | 地区 |
| Author | Varchar（255） | 作者 |
| Classfiy | Varchar（255） | 类别 |
| Create\_time | Datetime | 创建日期 |
| Description | Text | 简介 |
| Progress | Varchar(255) | 漫画进度 |
| Publishing\_house | Varchar(255) | 出版社 |
| Title | Varchar(255) | 标题 |
| Title\_img\_url | Varchar(255) | 漫画封面 |
| Update\_time | Datetime | 更新时间 |

#### t\_comic\_detail:用于存储漫画每一话的信息

表 3.3 数据库表t\_comic\_detail

|  |  |  |
| --- | --- | --- |
| **字段名称** | **字段类型** | **备注** |
| Id | Bigint | 漫画每话id |
| Name | Varchar（255） | 每话名称 |
| Urls | Varchar（255） | 每话图片图床链接集 |
| Create\_time | Datetime | 创建日期 |
| Comic\_id | Bigint | 外键到主表t\_comic |

#### t\_up\_comic:用于存储用户与其投稿漫画之间的联系

表 3.4 数据库表t\_up\_comic

|  |  |  |
| --- | --- | --- |
| **字段名称** | **字段类型** | **备注** |
| Id | Bigint | 投稿表id |
| Comic\_id | Bigint | 漫画id |
| User\_id | Bigint | 用户id |

#### t\_comic\_collect:用于存储用户与其收藏的漫画之间的联系，表结构与t\_up\_comic类似故不再详细赘述

#### t\_comic\_history:用于记录用户浏览过的漫画，方便用户下次阅读时能够快速找到

表 3.5 数据库表t\_comic\_history

|  |  |  |
| --- | --- | --- |
| **字段名称** | **字段类型** | **备注** |
| Id | Bigint | 历史表id |
| Comic\_id | Bigint | 漫画id |
| User\_id | Bigint | 用户id |
| Comic\_detail\_id | Bigint | 漫画话数id |
| Create\_time | Datetime | 创建时间 |
| Update\_time | Datetime | 更新时间 |

#### t\_classfiy\_info:用于存储漫画类别，方便日后的扩充

表 3.6 数据库表t\_classfiy\_info

|  |  |  |
| --- | --- | --- |
| **字段名称** | **字段类型** | **备注** |
| Id | Bigint | 漫画类别id |
| Classfiy\_name | Varchar(255) | 漫画类别名 |

#### t\_address\_info:用于存储漫画的地区，方便日后的扩充，库表结构与t\_classfiy\_info类似故不再赘述

#### T\_progress\_info:用于存储漫画的进度，方便日后的扩充，库表结构与t\_classfiy\_info类似故不再赘述

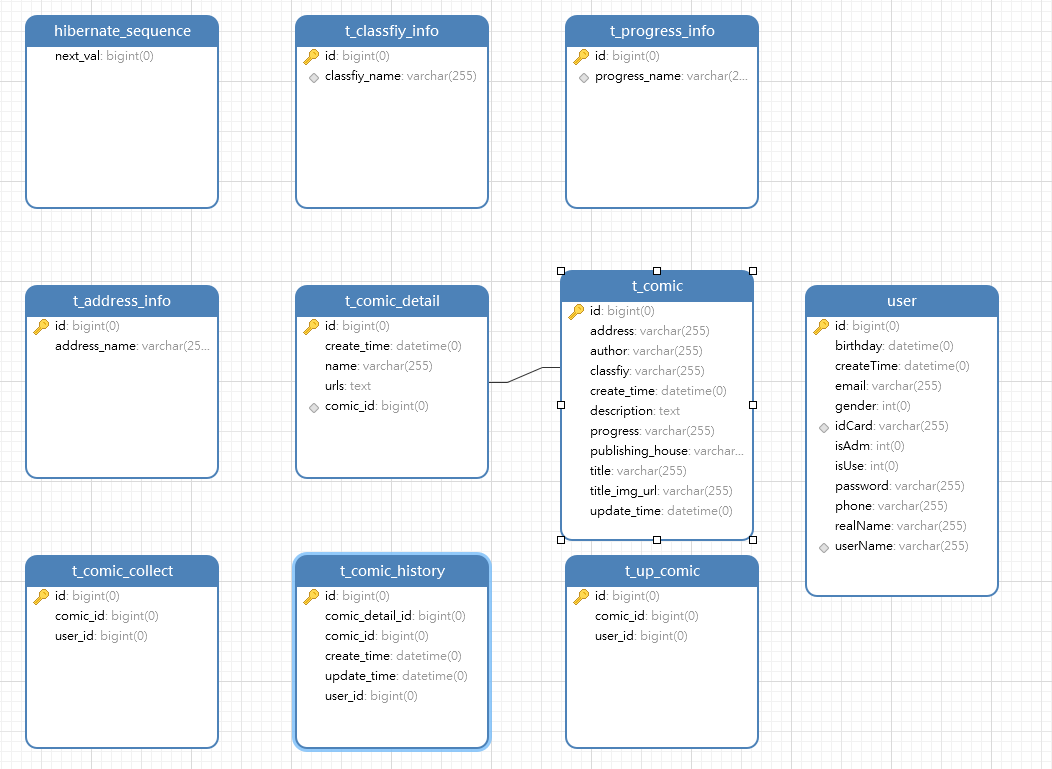


图 3.1 数据库整体ER图

系统详细设计与编码实现

系统详细设计与编码实现

# 系统详细设计与编码实现

## 程序结构

本系统后端采用三层架构，即dao层，service层以及controller层。在dao层中实现对数据库最简单的增删改查功能，并将数据库的数据封装为Java对象。在此层中本人使用了Spring家族中的Spring Data JPA这一持久化框架对数据库进行操作。通过继承JpaRepository接口，使我在开发过程当中几乎不需要再去书写Sql语句。通过框架，我们只需要在定义方法名的时候根据框架格式进行书写，比如说findById，即可实现相应功能，大大提高了系统的开发效率；在service层当中对dao层获取到的数据进行处理；在controller层当中实现对页面具体请求的处理。在Spring Boot当中我们可以通过添加不同的注解实现对应不同的功能。比如，仅仅通过在类或者方法上添加RequestMapping注解，使得前台页面跳转的链接将会被对应的方法获取到，以便我们进行数据的处理。不在像以前还需对其进行配置文件的书写。通过在类上添加RestController注解，或者在方法上添加ResponseBody注解即可轻松实现对对象数据直接以Json或者xml形式写入HTTP相应中，不再像以前的传统Spring MVC应用那样需要返回一整个视图，实现了前后端的分离。一方面减轻了系统开发的复杂程度，实现让一人专注于后端的开发，另一个只需要关注与前端的页面就好。在另一个方面，后端也不需要像以前一样返回一整个页面的数据，可以只需要返回页面中某一功能所需要的数据，并且是可以Json格式传输回的，大大改善了前端页面用户的使用体验。

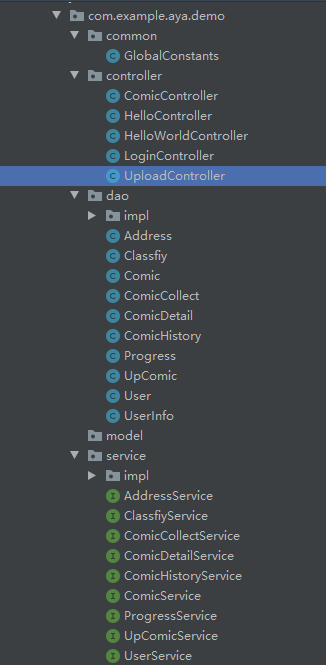


图 4.1 系统整体架构

## 系统注册功能的开发

### 系统注册功能前端代码的设计

系统的注册页面参考了BootStrap提供的登录案例进行改造，整体主要以一个form表单为主，在填写完成之后通过点击注册按钮以post的方式提交将表单内数据传入后端。在后端进行数据的校验工作，如果存在错误，将会再次返回到这个页面，由于整个页面的体量较小，不存在大量数据的传递，因此在设计之初也就没有进行前后端的分离，通过传统Spring MVC将整个视图进行返回。返回时通过Thymeleaf模板引擎对页面进行渲染。比如在最后一段当中进行了if表达式的判断，如果后端输出的数据当中带有errorMsg的话，就会显示div标签中的信息。此外，通过BootStrap框架，只需在class中输入对应的参数，比如glyphicon-user，就可以实现在input输入框中添加相应的图片。而Glyphicons Halflings 一般是收费的，但是他们的作者允许 Bootstrap 免费使用。

<form class="form-signin" th:action="@{/user/regist}" action="/user/regist" method="post" >

<h3 class="form-signin-heading">注册账号</h3>

<div class="form-group has-feedback">

<label for="inputUsername" class="sr-only control-label">用户名</label>

<input type="text" id="inputUsername" name="userName" class="form-control " placeholder="用户名" required="" autofocus="">

<span class="glyphicon glyphicon-user form-control-feedback" aria-hidden="true" ></span>

</div>

<!-- ................... -->

<div class="alert alert-danger" role="alert" th:if = "${errorMsg != null}" th:text="${errorMsg}"></div>

<button class="btn btn-lg btn-primary btn-block" type="submit">注册</button><br>

<a th:href="@{/user/toLogin}">已有账户，点击登录</a>

</form>

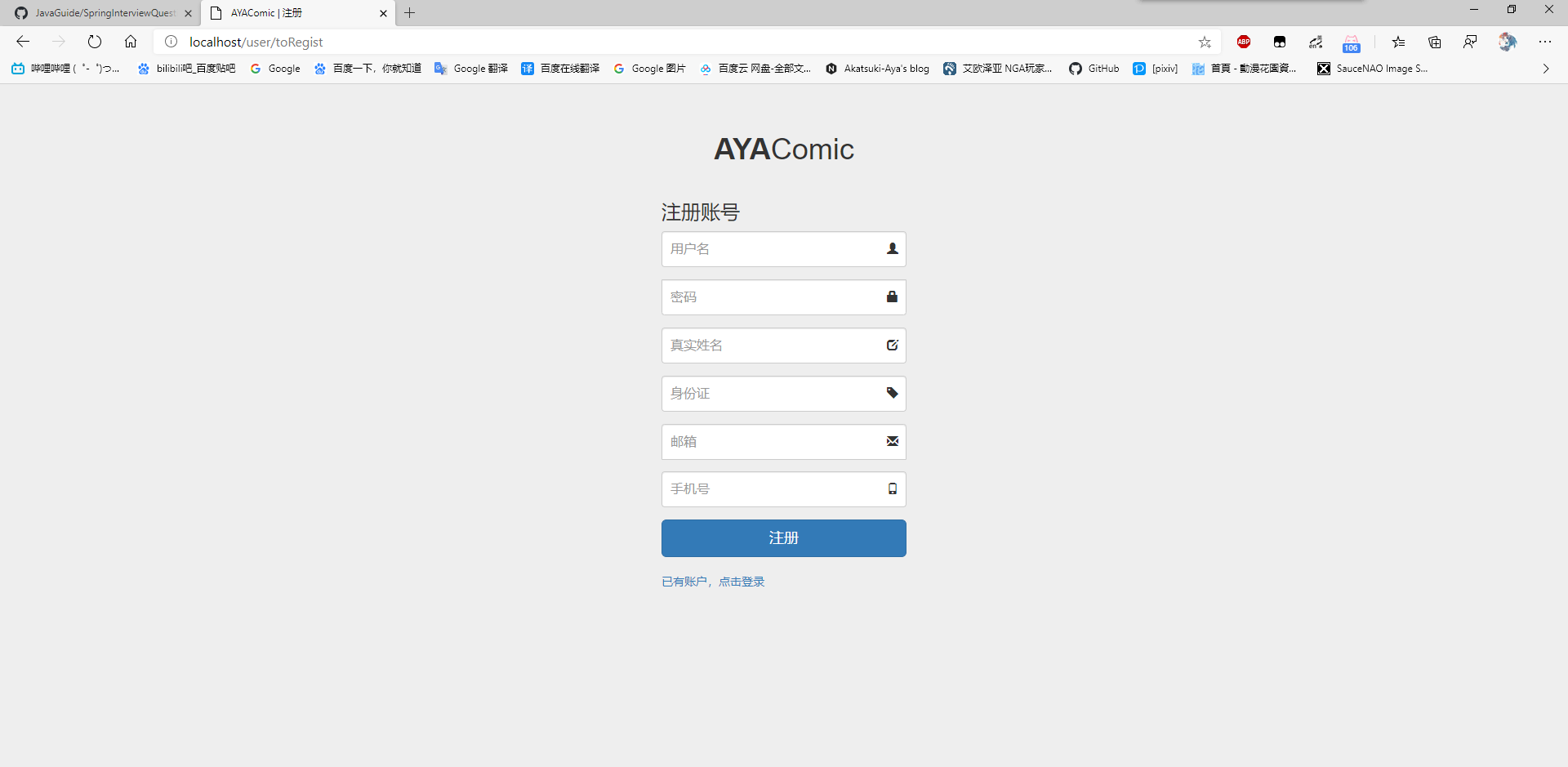


图 4.2 注册界面

### 系统注册功能后端具体业务实现

后端注册方法整体在LoginController类当中，通过在类和方法上添加RequestMapping注解，当页面跳转到/user/regist时，将自动被此controller的该方法获取到，并执行其代码。另一方面，通过在类中属性上添加Autowired注解，通过Java的反射机制将属性的值进行依赖注入。类似于设计模式当中的工厂模式，不需要我们再对对象自己进行创建初始化，而是在项目启动时，先将这类型的对象初始化好后，存入IOC容器当中，当我们需要用到其中某个对象时，只需从容器拿取就好。而IOC就像是个map集合，只要我们给它一个Key值就会返回对应的对象给我们。至此省去了我们往后对于此类对象的创建问题。降低了程序之间的耦合性和开发难度，大大增加了项目的可维护性。在最后如果注册成功将跳转到登录页面，若注册失败则会仍返回注册界面。

@Autowired

private UserService userService;

@RequestMapping("/regist")

public String regist(Model model, String userName, String password, String realName, String idCard, String phone, String email) {

User user = userService.regist(model,userName, password, realName, idCard, phone, email);

Object errorMsg = model.getAttribute("errorMsg");

if (errorMsg != null) {

return "/userPage/regist";

}

return "index2";

}

在userService接口的regist方法具体实现了注册的功能，通过阿帕奇lang3包下的StringUtils类的isBlank方法对表单中的各个字段进行非空判断。IsBlank相比传统的isEmpty方法是在此基础上进行了为空（字符串为空格，制表符，tab的情况）的判断，是更加安全，更加常用的一个方法。若为空，则在model中传入属性errorMsg及其值。

if (StringUtils.isBlank(userName)||StringUtils.isBlank(password)||

StringUtils.isBlank(realName)||StringUtils.isBlank(idCard)||

StringUtils.isBlank(phone)||StringUtils.isBlank(email)

) {

model.addAttribute("errorMsg","值为空");

return null;

}

然后通过CheckUtil类中的IDCardValidate，isMobileNO方法分别对表单中的身份证号和手机号的合法性进行验证，由于这两个方法都是在网上找到的由其他人已经写好了的现有的方法，我就拷贝直接进行使用了，在此就不再进行过多的赘述。

接着通过用户名或身份证对user表中的数据进行查询，保证此次的注册操作中，注册的用户在这次之前没有注册过，避免用户重复注册。最后通过DigestUtils类中的方法对表单中的密码数据进行md5加密，避免数据库数据泄露后，可以直接获取到用户的密码，保证了用户数据的安全性。至此，在数据库中添加新的user数据，并将注册好后的的数据在User类中进行封装并返回。

//验证用户名是否存在

User isUser = userimpl.findByUserNameOrIdCard(userName, idCard);

if (isUser != null) {

model.addAttribute("errorMsg", "用户名或身份证号已存在");

return null;

}

//密码加密

password = DigestUtils.md5DigestAsHex(password.getBytes());

//String encodePwd;

User user = new User(userName, password, realName, idCard, phone, email, 0);

try {

userimpl.save(user);

} catch (Exception e) {

model.addAttribute("errorMsg", e.getMessage());

}

## 系统登录功能的开发

### 系统登录功能前端代码的设计

系统登录总体的前端设计思路与注册界面基本类似，同样是以一个form表单为主，不再过多赘述，仅以图片展示页面。

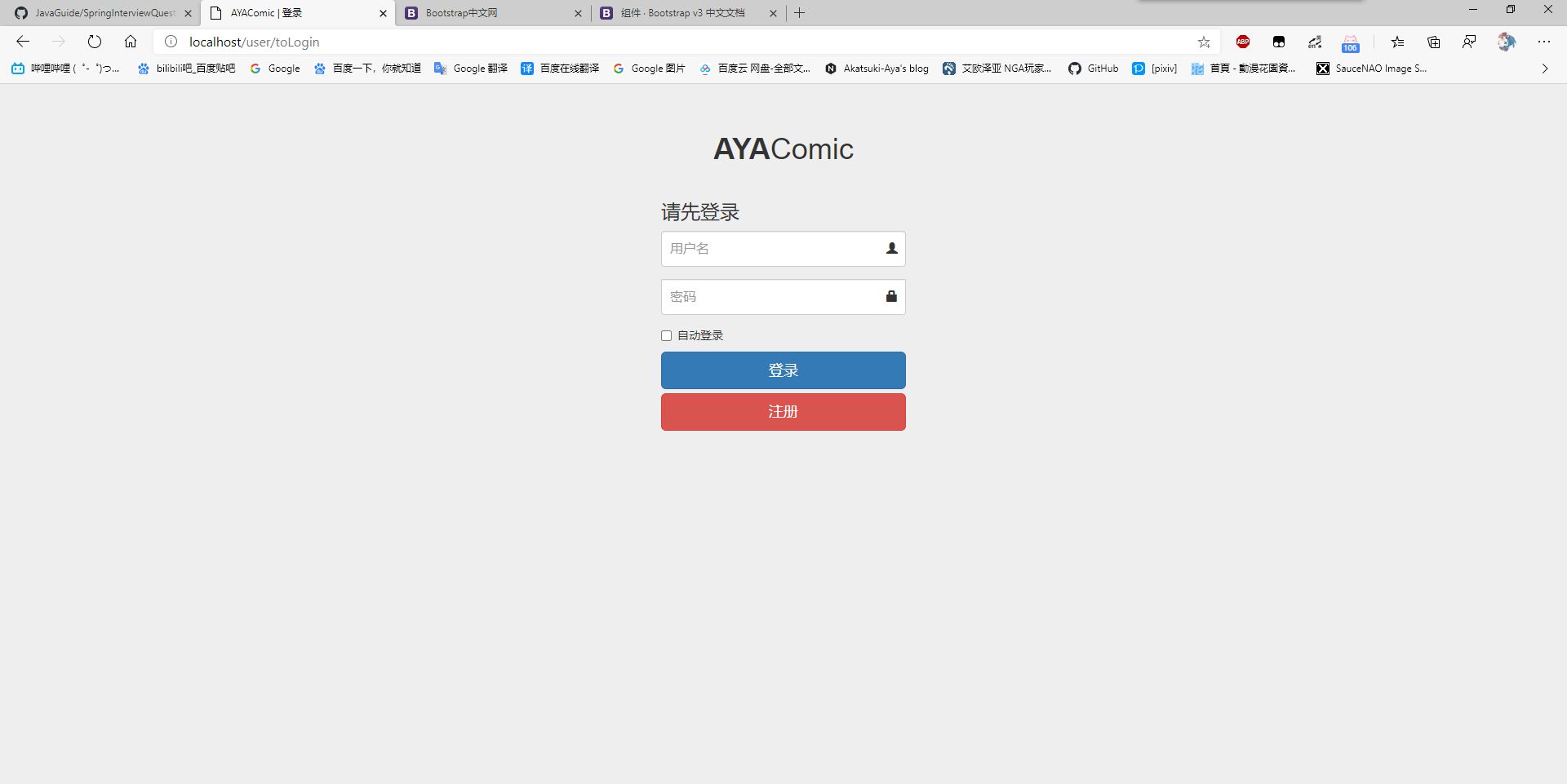


图 4.3 登录界面

### 系统登录功能后端具体业务实现

在登录功能中的部分与注册类似功能在此省略。Login方法中主要通过对登录表单中的密码也同样进行md5加密后与原数据库密码进行比对判断是否用户名和密码是否正确。然后将查询到user类中的userId添加到session中。使得userId保存在了服务器的会话当中，即使用户进行页面的跳转，会话中的变量也不会消失，方便日后在使用其他功能时，可以进行用户是否登录的判断。当会话中存在userId的数据时，说明该用户已经登录过了，可以使用接下来的功能，如果没有的话，则会将页面跳转到登录的界面。

@RequestMapping("/login")

public String login(Model model,String username, String password) {

if (userService.login(username,password)) {

HttpSession session = request.getSession();

User user = userService.findUserByUserName(username);

session.setAttribute("userId",user.getId());

return "redirect:/upload/toUploadManage";

} else {

model.addAttribute("msgErrorFlag", false);

model.addAttribute("errorMsg", "用户名或密码出错");

return "index2";

}

}

## 漫画投稿功能的开发

### 漫画投稿功能前端代码的设计

首先就投稿整体的前端页面进行展示。

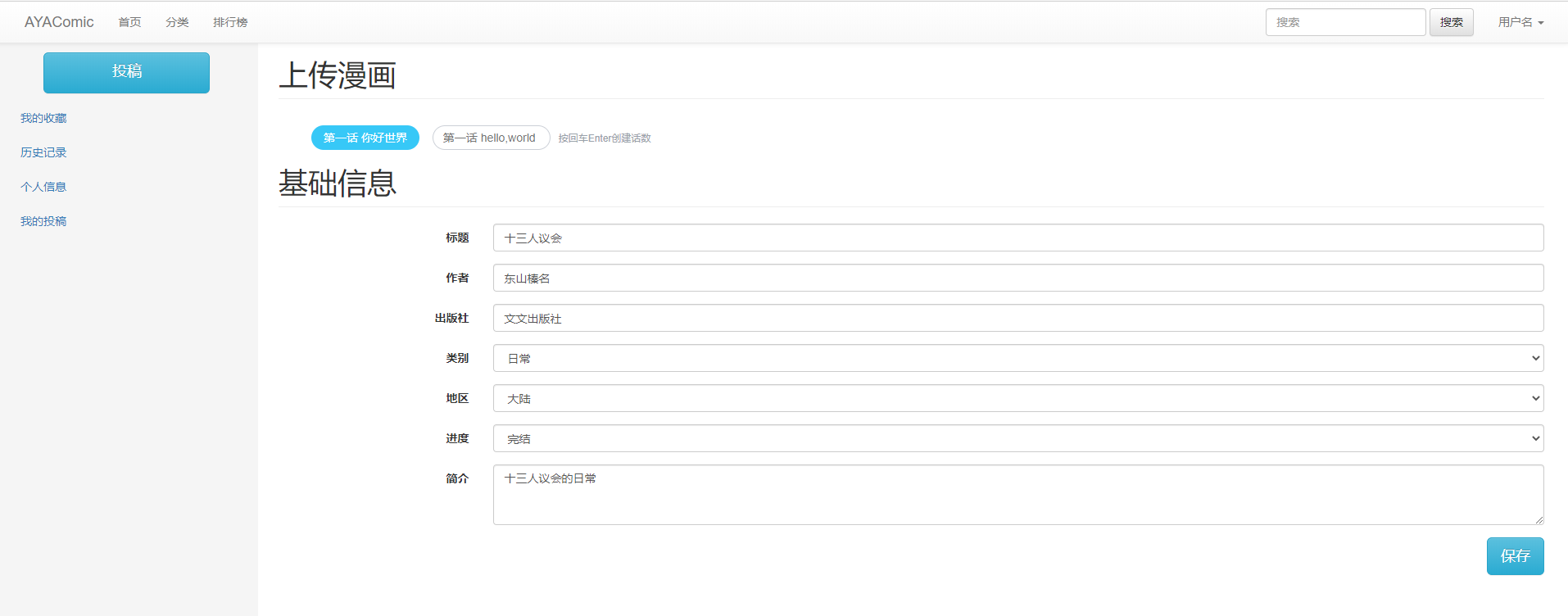


图 4.4 漫画投稿界面

整个投稿界面前端主要涉及四大功能。每一话的创建与删除并且可以通过输入框自定义每一话的名称；通过使用github上的Krajee插件实现前端多图片异步上传至后端然后回调返回上传结果数据；最后的保存，将上传漫画和基础信息两块的信息全部传入后端进行保存；如果是对已经投稿的漫画进行编辑则对漫画内容进行显示初始化。

#### Krajee插件与文件上传

Krajee是一款在github上开源的，以BootStrap前端框架为基础制作的文件上传插件。插件本身提供了多种的主题样式供我们选择，并且提供了多样的属性参数供我们传入。提供了像是文件拖拽，生成预览图，显示文件上传进度，文件上传完毕后自动刷新预览区域等诸多功能。插件提供了图片全部上传完毕后的回调函数，我通过将后台图片上传至七牛云后，将每张图片的链接存入一个集合当中，然后统一以Json数据的方式传回，再回调后将对应html元素添加一uped\_urls属性，并将Json结果集赋值于它。

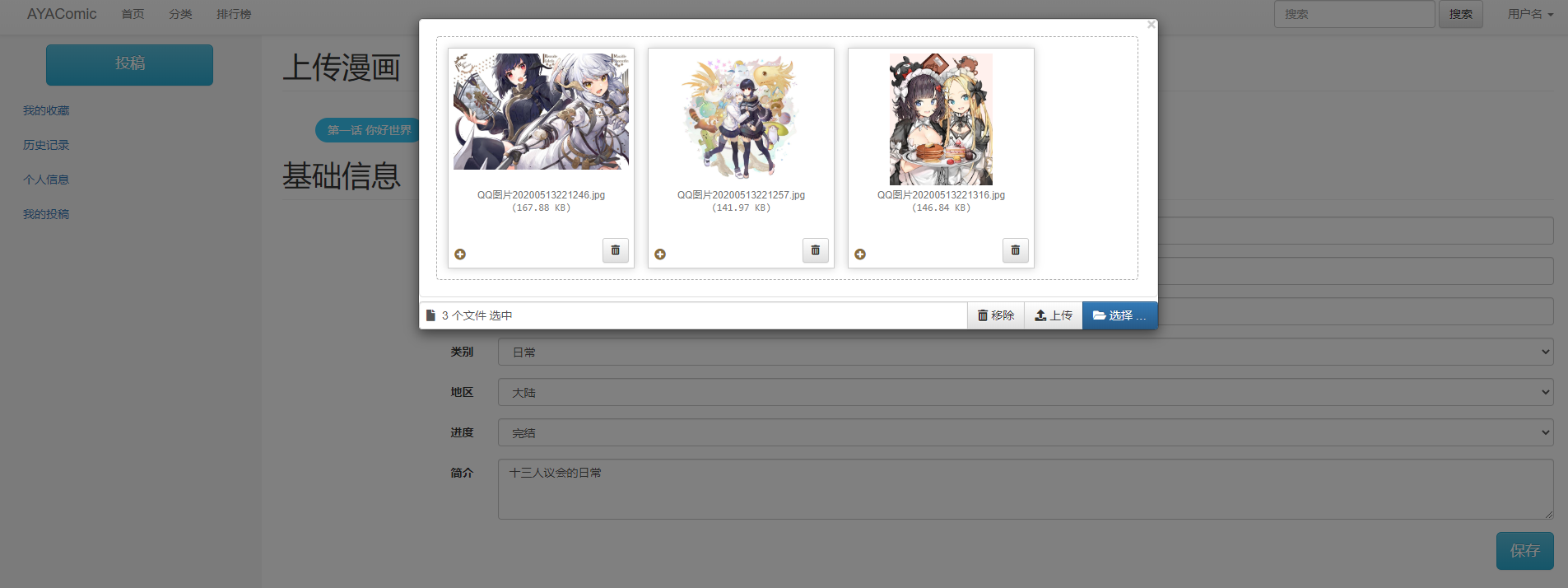


图 4.4 Krajee插件

以下代码将以注释的形式展示该插件主要参数的各项功能作用。

function initFileInput(name,urls) {

var id= name.substring(11);

var id="upload-"+id;

$("#"+name).fileinput({

theme: 'fas',//设置主题

language: 'zh', //设置语言

uploadUrl: "upFile", //上传的地址

allowedFileExtensions: ['jpg','png'],//接收的文件后缀

uploadAsync: false, //默认异步上传

showUpload: true, //是否显示上传按钮

showRemove: true, //显示移除按钮

showPreview: true, //是否显示预览

showCancel:true, //是否显示文件上传取消按钮。默认为true。只有在AJAX

上传过程中，才会启用和显示

dropZoneEnabled: true,//是否显示拖拽区域

/\*minImageWidth: 50, //图片的最小宽度

minImageHeight: 50,//图片的最小高度

maxImageWidth: 1000,//图片的最大宽度

maxImageHeight: 1000,//图片的最大高度

maxFileSize: 1024,//单位为kb，如果为0表示不限制文件大小

minFileCount: 1, //每次上传允许的最少文件数。如果设置为0，则表示文件数

是可选的。默认为0

maxFileCount: 0, //每次上传允许的最大文件数。如果设置为0，则表示允许的

文件数是无限制的。默认为0\*/

enctype: 'multipart/form-data',

}).on('fileuploaded', function(event, data, previewId, index)

{ //异步上传完成

}).on('filebatchuploadsuccess',function (event,data,files,extra)

{ //同步上传完成

var myLi = $("#"+id);

myLi.attr("uped\_urls",data.response.fileUrls);

mydata = $("#"+id);

});

}

#### BootStrap模糊框插件

通过button标签中的data-toggle参数指定该按钮用于控制一个模糊框，并通过data-target属性加模糊框与按钮进行绑定。通过 aria-describedby 属性为模态框 .modal 添加描述性信息。添加 role="dialog" 和 aria-labelledby="..." 属性，用于指向模态框的标题栏；为 .modal-dialog 添加 aria-hidden="true" 属性。

<button type="button" class="btn btn-primary btn-lg" data-toggle="modal" data-target="#myModal">

Launch demo modal

</button>

<!-- Modal -->

<div class="modal fade" id="myModal" tabindex="-1" role="dialog" aria-labelledby="myModalLabel">

<div class="modal-dialog" role="document">

<div class="modal-content">

<div class="modal-header">

<button type="button" class="close" data-dismiss="modal" aria-label="Close"><span aria-hidden="true">&times;</span></button>

<h4 class="modal-title" id="myModalLabel">Modal title</h4>

</div>

<div class="modal-body">

...

</div>

<div class="modal-footer">

<button type="button" class="btn btn-default" data-dismiss="modal">Close</button>

<button type="button" class="btn btn-primary">Save changes</button>

</div>

</div>

</div>

</div>

#### 每一话的创建

通过对class属性为comic-num的input标签添加键入监控，当键入回车时，创建li标签元素并初始化一个文件上传插件和一个模糊框插件与其绑定，插入至addItem元素之前。最后为其添加鼠标相关事件。在此功能中，通过使用jQuery框架中的类选择器和find方法实现了对元素的快速定位获取。通过clone方法复制demo元素，实现对对象的快速创建。

<li class="add-item">

<input type="text" maxlength="20" class="comic-num" placeholder="

第一话 hello,world">

<span class="tip">按回车Enter创建话数</span>

</li>

$(".comic-num").bind('keypress',function(event){

if(event.keyCode == 13)

{

var text = $(".comic-num").val();

text =$.trim(text);

if( text==""){

return;

}

var addItem = $(".add-item");

var newItem= $("<li></li>").text( text);

$(".comic-num").val("");

newItem.addClass("comic-num-item ");

newItem.attr("modalId","upload-"+i.toString());

newItem.attr("id","upload-"+i.toString());

addItem.before(newItem);

var upload= $(".upload-damo").clone();

upload.attr("class","modal fade upload-"+i.toString());

var uploadImg= upload.find("input");

var uploadImgIdName="upload-Img-"+i.toString();

uploadImg.attr("id",uploadImgIdName);

$(".upload-list").append(upload);

initFileInput(uploadImgIdName);

addMouseEvent();

i++;

}

});

#### 鼠标监控事件与话的删除

在鼠标监控事件中主要用于监控鼠标进入和鼠标移出两个事件，当鼠标进入其中一话的html元素当中时，为其添加一个关闭图标，并通过修改class属性的方式改变元素的css样式，达到鼠标进入时的视觉可视化效果。并通过给文字部分和关闭图标部分，两个部分不同的click功能实现点击文字时出现模糊框效果，点击关闭时将会删除整个元素。而话的删除则是通过remove方法将元素，十分简单，并调用deteleComicDetail方法，传入漫画话数id，通过Ajax异步调动后代代码将漫画对应话数从数据库中删除。



图 4.5 鼠标进入元素

function addMouseEvent(){

$(".comic-num-item").off('mouseenter');

$(".comic-num-item").off('mouseleave');

$(".comic-num-item").mouseenter(function(e){

var oldText = $(this).text()+" ";

$(this).text("");

var oldItem= $("<span></span>").text(oldText);

oldItem.addClass("Item-move-on");

oldItem.attr("data-toggle","modal");

var target =$(this).attr("modalId");

oldItem.attr("data-target","."+target);

var newItem= $("<span></span>");

newItem.addClass("glyphicon glyphicon-remove");

newItem.attr("aria-hidden","true");

newItem.click(function(e){

var modalId = $(this).parent().attr("modalId");

var detailid = $(this).parent().attr("detailid");

if (detailid != null){

deteleComicDetail(detailid);

}

$("."+modalId).remove();

$(this).parent().remove();

});

$(this).append(oldItem,newItem);

});

$(".comic-num-item").mouseleave(function(e){

var oldText = $(".Item-move-on").text();

oldText = $.trim(oldText);

$(this).text(oldText);

});

}

#### 编辑漫画的初始化

编辑漫画时的初始化主要是通过Thymeleaf模板引擎对页面进行渲染。Js代码部分主要是对页面中的各个元素添加事件监控。并且由于Krajee插件只能通过Js代码进行初始化，通过带入urls参数，为文件上传插件赋入初始图片。

var i =0;

init();

//初始化 mouse按钮

addMouseEvent();

function init(){

var length = $(".comic-num-list").children().length-1;

for(i ; i<length;i++){

var upload= $(".upload-damo").clone();

upload.attr("class","modal fade upload-"+i.toString());

var uploadImg= upload.find("input");

var uploadImgIdName="upload-Img-"+i.toString();

uploadImg.attr("id",uploadImgIdName);

$(".upload-list").append(upload);

var urls = $("#upload-"+i.toString()).attr("uped\_urls");

urls=JSON.parse(urls)

initFileInput(uploadImgIdName,urls);

}

}

#### 保存

通过将serialize方法将漫画基础信息表单中的数据序列化为字符串，然后将每一话的数据以JSon的形式存储添加到comicList列表中，最后将其json字符串话拼接到data中通过Ajax异步上传到后代中。

function save() {

var data = $("form").serialize();

var liList = $(".comic-num-list").children();

var comicList =[];

for(var i=0; i<liList.length-1; i++){

var text = $(liList[i]).text();

var urls = $(liList[i]).attr("uped\_urls");

var id = $(liList[i]).attr("detailid");

if(urls!=null && urls !=""){

var list ={name:text,urls:urls,id:id};

comicList.push(list);

}

}

var comicJson = JSON.stringify(comicList);

data=data+"&comicJson="+comicJson;

$.ajax({

type:"POST",

url:"addComic",

data:data,

async:true, // 异步，默认开启，也就是$.ajax后面的代码是不是跟$.ajx里面的代码一起执行

dataType:"json", // 返回浏览器的数据类型，指定是json格式，前端这里才可以解析json数据

success:function(data){

alert("上传成功");

},

error:function() {

alert("error");

}

});

}

### 漫画投稿功能后端具体业务实现

#### 进入投稿页面前

首先通过session会话中的userId判断用户是否已经登录，如果没有登录的话跳转回登录页面，然后通过是否传入comicId参数判断是投稿一个新的漫画，还是对原来的漫画进行编辑。在getAllClassifyAddressProgress方法中为model添加类别，进度，地区的集合，传入前台下拉框当中。

@RequestMapping("/toUploadImgFile")

public String toUploadImgFile(Model model, Long comicId) {

if (!checkIsLogin()) {

return "redirect:/user/toLogin";

}

Comic comicById = new Comic();

if (comicId != null) {

comicById = comicService.findComicById(comicId);

}

model.addAttribute("comic",comicById);

this.getAllClassifyAddressProgress(model);

return "/userManage/comicUpload";

}

通过对两个POJO类添加对应的OneToMany和ManyToOne注解实现在数据库查询t\_comic表后将数据封装到Comic类对象时将会自动将对应t\_comic\_detail表中的数据封装并添加到comicDetailList集合当中。

//Comic类

@OneToMany(mappedBy = "comicId",cascade=CascadeType.ALL,fetch=FetchType.EAGER)

private List<ComicDetail> comicDetailList;

//ComicDetail类

@ManyToOne(cascade={CascadeType.MERGE,CascadeType.REFRESH},optional=false,fetch = FetchType.LAZY)

@JoinColumn(name = "comic\_id")

private Comic comicId;

#### 图片上传至七牛云接口

通过RequestParam注解获取前台的图片数据并封装到Spring框架提供的MultipartFile数组当中。接着通过七牛云提供的接口将图片数据以字节的方式传入参数当中，上传成功后返回一个对应的url链接将其添加到urls对象的list集合当中，通过阿里的FastJson转换为json字符串，并通过一个名为result的json对象传回前台。

@ResponseBody

@RequestMapping(value = "/upFile",method= RequestMethod.POST)

public String upFile(@RequestParam("file\_data") MultipartFile[] files)throws Exception{

//.....

String fileId = request.getParameter("fileId");

List<String> urls = new ArrayList<>();

for (MultipartFile file:files) {

byte[] bytes = file.getBytes();

String fileUrl = QiNiuUtil.upload(bytes);

urls.add(fileUrl);

}

JSONObject result = new JSONObject();

result.put("msg","success");

result.put("fileUrls",JSONObject.toJSONString(urls));

return result.toJSONString();

}

#### 漫画投稿

在SpringBoot当中如果形参为自定义类时，如果前台请求出去的数据name与自定义类中的属性的名相同时，将会自动将数值封装到类当中。如此使得在漫画基础信息表单中的数据直接封装到了Comic类型的comic变量当中去了。省去我们挨个依次赋值的步骤。然后通过前台传入的参数当中是否含有id判断出此次保存是新增操作还是修改操作。如果id为空就会为其设置创建时间等属性，如果不为空则为其设置更新时间的属性。接着在通过Spring Data Jpa框架的帮助下，框架提供了save方法，可以根据传入变量的属性判断是对其执行更新操作还是创建操作。并且由于在Comic类当中添加了GeneratedValue注解，并设置其类型为IDENTITY也就是自增，使得保存的数据id即使为空，在存入数据库当中时也能够在前表数据的基础上累积自增，并将存入后的数据继续以Comic类进行封装并返回。使得我们不管是重新编辑的漫画还是新投稿的漫画，我们都拥有了漫画的id值，方便我们再对前端上传漫画表单中的上传至图传后返回的链接集以及每一话的基础信息能够和他们所对应的具体漫画得以联系。在通过阿里fastJson中parseArray方法快速将json字符串转换为json数组。通过for循环与fastjson中的getObject方法快速将一个json对象数据封装为ComicDetail类型的数据。然后依次将它们存入数据库当中。最后再实例化一个UpComic对象，将上传用户与投稿的漫画建立连接，存入到t\_up\_comic表中去。

@RequestMapping(value = "/addComic", method = RequestMethod.POST)

public String addComic(Comic comic, String comicJson) {

//漫画添加

boolean isNew = true;

if(comic.getId()!= null){

isNew =false;

Comic comicById = comicService.findComicById(comic.getId());

if (comicById!= null){

comic.setCreateTime(comicById.getCreateTime());

}

//....

comic.setUpdateTime(new Date());

Comic savedComic = comicService.saveComic(comic);

System.out.println(savedComic.getId());

//漫画详细添加

JSONArray comicJsonList = JSONObject.parseArray(comicJson);

for (int i = 0; i < comicJsonList.size(); i++) {

ComicDetail comicDetail = comicJsonList.getObject(i, ComicDetail.class);

if (comicDetail.getId()!=null){

ComicDetail comicDetailByIdAndComicId = comicDetailService.findComicDetailByIdAndComicId(comic.getId(), comic);

//......

comicDetail.setComicId(savedComic);

comicDetailService.saveComicDetail(comicDetail);

}

//up漫画联系添加

if(!isNew){

}else {

HttpSession session = request.getSession();

Long userId = (Long)session.getAttribute("userId");

Long comicId = savedComic.getId();

UpComic upComic = new UpComic(userId,comicId);

upComicService.saveUpComicService(upComic);

}

//....

}

## 我的投稿前台管理功能的开发

### 我的投稿前台管理页面的设计

在我的投稿页面当中主要通过Bpage插件实现了前台动态的分页功能。由于精力有限，本人自己所能录入的数据量较少，为了使得整体页面能够呈现出分页功能的效果，所以将pageSize即每页返回的漫画数设置为了1。

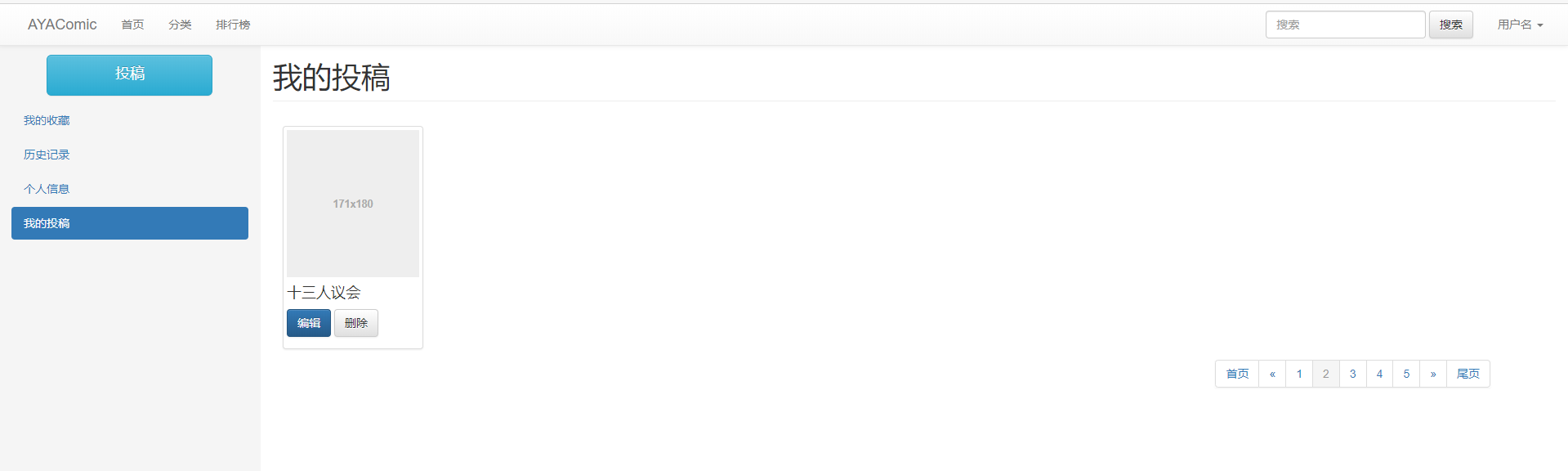


图 4.6 我的投稿页面

#### Bpage插件

Bpage插件也是我在网上找到的一款基于JQuery开发，用于BootStrap3环境下的前端独立分页插件。插件提供了页面跳转模式，异步请求页面模式（服务端页面），异步请求数据模式（服务端JSON数据）三种模式，使得我们可以随意挑选其中任意一种适合的进行开发项目。在本次开发过程中我所使用的第一种模式。Bpage插件主要由HTML和JS两部分组成。在HTML当中四个隐藏的input输入框将后台的分页数据存入当中。pageNumber用于指定分页当中的当前页数；pageSize用于指定分页当中的每页显示的数据量；totalPage用于指定总共多少页；totalRow用于指定总数据量；在js代码中通过id选择器获取这些值来初始化分页插件。

//Bpage插件HTML部分

<div class="paging" style="margin-right: 80px;">

<input type="hidden" id="pageNumber" th:value="${currentPage}">

<input type="hidden" id="pageSize" th:value="${1}">

<input type="hidden" id="totalPage" th:value="${page.totalPages}">

<input type="hidden" id="totalRow" th:value="${page.totalPages}">

<div id="page1"></div>

</div>

//Bpage插件JS部分

$("#page1").bPage({

//页面跳转的目标位置

url : 'toUploadManage',

//分页数据设置

totalPage : $('#totalPage').val(),

totalRow : $('#totalRow').val(),

pageSize : $('#pageSize').val(),

pageNumber : $('#pageNumber').val(),

//页面跳转时需要同时传递给服务端的自定义参数设置

params : function(){

return {

//userName : 'Akatsuki\_Aya',

//age : 22

};

}

});

#### 漫画样式设计

因为每一本漫画整体展示下来的内容大差不差，有所变化的无非是漫画的标题，封面等等的内容。因此使用了Thymeleaf中的th:each对comicList集合进行foreach循环渲染。通过th:attr自定义标签的属性，在div标签中存入了一个隐藏数据comicId，方便我们在对漫画进行编辑或者删除操作的时候，在点击按钮后向上查询父节点元素获取comicId的值作为参数传入后台来定位具体的漫画。在另一方面，通过overflow，text-overflow，white-space等css样式属性来限制漫画标题的长度，防止过长的漫画标题破坏掉整体的页面样式布局。将超出部分以省略号的形式展示出来，然后通过设置title属性，让鼠标移动到标题上时能够弹出提示，依然可以看到完整的标题内容。

<div th:each="comic,iterStat : ${comicList}">

<div class="thumbnail" th:attr="comicId=${comic.id}">

<a onclick="">

<img data-src="holder.js/100%x180" alt="100%x180" style="height: 180px; width: 100%; display: block;" src=".." data-holder-rendered="true">

</a>

<h4 th:text="${comic.title}" th:attr="title=${comic.title}" style="overflow: hidden;text-overflow: ellipsis;white-space: nowrap">

漫画1

</h4>

<p><a onclick="toUpload(this)" class="btn btn-primary" role="button">编辑</a> <a class="btn btn-default" role="button">删除</a></p>

</div>

</div>

### 我的投稿后端具体业务实现

在本小节当中，对于上述已经提到过的普通的增删改查方法就不在进行过多的赘述了，主要就Springboot当中如何快速的实现分页功能进行阐述。在Spring Data Jpa框架当中，在继承JpaRepository接口后，通过在方法上加入Pageable形参，然后将方法的返回类型设置Page<T>类型，一个含有分页功能的查询方法就可以基本实现了！是不是特别的便捷。Pageable类其实就类似于sql语句当中limit对数据库表进行分页查询，该类型通过静态方法PageRequest.of就可以实例化，通过简单的传入page和size两个参数，就可以指定分页表中的当前页数和对分页的大小进行简单的设置，同时还可以通过静态方法Sort.by对数据库当中的指定字段进行排序。至此就可以对数据库数据进行分页查询了。然后将查询出来的数据通过Page类进行封装。在Page类对象中可以通过getTotalPages，getTotalElements等方法去获取分页查询出的分页数据。通过getContent方法将分页查询出的数据集合以List集合的方式返回出来。

Pageable pageable=null ;

if(pageNumber != null){

//PageRequest.of(page, size, Sort)

pageable = PageRequest.of(pageNumber - 1, 1, Sort.by(Sort.Direction.ASC, "id"));

model.addAttribute("currentPage",pageNumber);

}else {

model.addAttribute("currentPage",1);

pageable = PageRequest.of(0,1, Sort.by(Sort.Direction.ASC,"id"));

}

//Jpa具体接口 Page<UpComic> findByUserId(Long userId, Pageable pageable);

Page<UpComic> upComicPage = upComicService.findUpComicByUserId(userId, pageable);

## 分类页面功能的开发

### 分类页面的前台页面的设计

在分类页面的前端页面设计过程当中，主要涉及到了前台动态分页功能的实现和将漫画表中的分类，进度，地区的id值转换为具体名称。分页功能之前的章节中已经说过了，这样同样也就不再过多的废话了。此页面的重点功能主要集中在后端的多条件动态查询上。就简单说明前端的部分细微功能。首先因为在此页面当中我们可以对题材，地区，进度进行个性化的自由选择，但当我们选择分页后为了防止选择的条件消失，用到了之前在Bpage插件中提到过的params属性将我们所选的条件封装为json数据后连同分页数据一同传入至后端当中。另外一点，由于后端返回的漫画数据无论是在数据库表中还是后端类中，对于题材，进度，地区的存储都是存储它们的id值，因此直接返回后如果不加处理的显示出来的也是id数字，这样显然是不行的。好在在这一页当中本来也就需要返回这3类的一整个集合列表供用户选择。因此就不在后端中对漫画数据进行转换，而是通过传入前后后通过js代码进行初始化，将这3类分别放入到对应的span标签中，通过js代码查找对应id的名称值来进行替换，以此来实现前端键值对的转换。在后来的其他页面当中，由于懒得再写后端转换了，于是也就通过将页面上的上方3个集合列表style的display属性对其进行隐藏，继续通过js的方式，实现对其的转换，不再后端重复实现相同的功能了。



图 4.7 分类页面

### 分类页面的后端具体业务实现

在分类页面当中，共有题材，进度，地区三个选项供用户选择，且不同选项之前可以多重组合。故在后端代码中必须针对此，实现多条件的动态查询功能。在本系统的设计当中，依然是通过了jpa框架继承JpaSpecificationExecutor接口来实现此功能。在service层中通过lambda表达式来实现功能。Lambda 表达式是 Java8版本中发布的一个十分重要的新特性，可以取代大部分的匿名内部类，写出更优雅的 Java 代码。由于lambda表达式的特性，我们在编写的时候不需要在声明参数类型，编译器可以统一识别参数值，当参数只有一个时无需定义圆括号，主体如果只有一条语句，甚至连大括号可以省略掉，且表达式的值编译器会自动返回。使得整体的代码非常简洁，非常容易进行并行计算，突出了原有匿名内部类中真正有用的那部分的代码。说回多条件动态查询，首先我通过将条件整合封装到一个Comic类型的对象当中，在lambda表达式当中创建了一个Predicate类型的List集合。然后通过CriteriaBuilder这一工厂类，来创建安全查询的criteriaQuery对象，该对象是用来构建查询的，添加到predicates集合当中。在最后通过criteriaQuery对象的where方法，和predicates对象的toArray方法形成一条完整的sql供程序去执行查询数据。

Public interface ComicImpl extends JpaRepository<Comic, Long>, JpaSpecificationExecutor<Comic>

public Page<Comic> findByCondition(Integer page, Comic comic) {

Pageable pageable = PageRequest.of(page, 2);

return comicImpl.findAll((root, criteriaQuery, criteriaBuilder) -> {

List<Predicate> predicates = new ArrayList<Predicate>();

if(comic.getId()!=null){

predicates.add(criteriaBuilder.equal(root.get("id"),comic.getId()));

}

//......

if (comic.getTitle() != null && !"".equals(comic.getTitle())){

predicates.add(criteriaBuilder.like(root.get("title"),"%"+comic.getTitle()+"%"));

}

Return criteriaQuery.where(predicates.toArray(

new Predicate[predicates.size()])).getRestriction();

},pageable);

}

## 实体漫画浏览功能的开发

### 实体漫画浏览页面的前台页面的设计

在实体漫画浏览页面的前台设计当中，主要用到了模糊框插件，通过JQuery和Ajax实现对漫画收藏功能以及历史记录功能。在css样式中通过设置：hover属性的样式，实现当鼠标移动到按钮上时会呈现出不同的颜色，达到视觉可视化的效果，增强用户的体验。通过将标题作者简介等HTML元素设置display: block属性，将此元素将显示为块级元素，使此元素前后会带有换行符。同样的对于章节列表通过设置display: -ms-flexbox;display: flex;align-items: center;overflow: visible;等属性实现按钮的自动换行，当按钮的长度超出了div规定的宽度时，会自动转换到下一行，从而减轻前台代码判断的压力，不用通过js代码或者是模板引擎来进行判断。



图 4.8 实体漫画浏览页面



图 4.9 通过模糊框浏览漫画

当点击收藏按钮时，会调动js中的changeCollect函数通过获取在漫画标题所在的HTML元素中隐藏加入了comicId属性获取漫画的id值，然后经过Ajax异步调度后台代码实现用户对漫画的收藏。对于历史记录的功能与其十分类似，无非是把点击收藏按钮换成了点击各个章节，并且在各个章节中也暗藏了comicDetailId值。在后端完成更新保存。并且在异步上传前，修改了开始阅读按钮中的historyId属性，使得在页面中点击开始阅读时，也能够直接阅读到最近一次阅读的内容中去。而对于漫画内容的展示，就想前文说的，是用到了BootStrap的模糊框插件，但与之前不同的是，之前是一个按钮绑定一个模糊框，而在这个页面当中是多个按钮绑定一个模糊框。是的，自始至终所用到的模糊框都是那一个，通过在每一个章节按钮中添加urls属性暗中为每个按钮添加了对应章节的图片链接集合数据。当每一次点击其中一个按钮时，调用showComic函数，首先先清空模糊框中的所有子元素，再将按钮中的图片集合，通过for循环以子节点的方式依次添加至模糊框当中，实现点击不同章节显示不同漫画的效果。

function changeCollect() {

var comicId = $(".comic-title").attr("comicId");

$.ajax({

type:"POST",

url:"addComicCollect",

// data:{"id":val},

data:{comicId:comicId},

async:true,

dataType:"json",

success:function(data){

mydata=data;

$(".isCollect").text(data.isCollect);

//alert("上传成功");

},

error:function() {

alert("error");

}

});

}

### 实体漫画浏览页面的后端具体业务实现

此部分的后端代码与上方存在些许相似之处，且漫画收藏与漫画历史记录保存功能的后台功能实现也具有十分相似的地方，因此我仅挑选出漫画收藏部分进行简要的说明。首先因为前台是通过Ajax异步调用的，因此采用前后端分离的方式，通过在方法上添加ResponseBody注解实现方法返回json数据回前端页面，这里我所使用了阿里的fastjson来返回json对象的json字符串。前台通过后代json中的isCollect属性的数值，改变收藏按钮的文字内容。通过用户的id和漫画id查看之前数据库当中是否有存过数据，以此作为依据来判断用户是否收藏过此本漫画。

@ResponseBody

public String addComicCollect(Long comicId){

HttpSession session = request.getSession();

Long userId = (Long)session.getAttribute("userId");

ComicCollect comicCollect =

comicCollectService.findComicCollectByUserIdAndComicId(userId, comicId);

JSONObject result = new JSONObject();

if(comicCollect != null){

comicCollectService.deleteComicCollectById(comicCollect.getId());

result.put("msg", "取消收藏");

result.put("isCollect", "收藏");

}else {

comicCollect = new ComicCollect();

comicCollect.setComicId(comicId);

comicCollect.setUserId(userId);

comicCollectService.saveComicCollect(comicCollect);

result.put("msg", "收藏成功");

result.put("isCollect", "已收藏");

}

return result.toJSONString();

}

## 首页功能的开发

### 首页页面的前台页面的设计

首页当中的核心功能点为巨幕的制作，通过BootStrap中的巨幕插件进行实现。核心功能点与模糊框想类似，通过为ol元素的class属性添加carousel-indicators值，使其成为一个巨幕，然后通过li元素中的data-target属性与巨幕中的元素进行绑定。漫画的跳转与之前所有页面中漫画的跳转都相类似就不说了。



图 4.10 首页

### 首页页面的后端具体业务实现

在后端功能中我主要说明下Query注解，依旧是十分强大的Springboot框架带来的功能通过在底层方法上加上注解，就可以实现对数据库表的查询。值得注意的是，Query注解中的并非sql语句而是hql语句。此语句的作用是查询t\_comic\_collect表以comicId进行分组后根据每组comicId的数量进行降序排序以达到排行榜的效果，然后通过分页功能实现只取前10，图中由于数据量实在是太少所以只取到了3本漫画，如果放在真实的业务场景中不会出现这种情况。

@Query("select t from ComicCollect t \n" +

"GROUP BY t.comicId\n" +

"ORDER BY count(t.comicId) desc ")

Page<ComicCollect> getComicCollect(Pageable pageable);

# 结论

## 系统的不足

由于疫情的影响和自身的惰性，导致在整个毕设设计的前半段基本是在浑水摸鱼混日子，对于毕设没有做好一个充足的准备。而后半段由于实习的缘故导致事务繁重，导致最终用来开发系统的时间大约只有两周左右。通过一周的时间设计前端页面，另一周来实现系统的后端功能以及数据库表的设计。因此带来的最为明显的一点就是数据量的不足，没有充足的时间去录入庞杂的数据。也并没有足够的时间去对系统进行测试。对于当初设想的部分功能最后也没有时间去做完。比如说当初设想是可以再有一个评论功能的，在浏览具体漫画的最下方，用户可以留下自己的评论，碍于时间等方面的原因并没有对此进行实现，已经页面上的排行榜页面最后也没有制作完成，只在首页上实现了一个相似的功能，放弃了具体页面的制作。以及在漫画系统中部分细节的欠考量。比方说，在漫画一章节浏览到最后时，没有提供一个按钮点击前往下一章节继续阅读；没有对已阅读漫画章节进行样式设计，可以让用户辨认出那些章节用户是阅读过的。

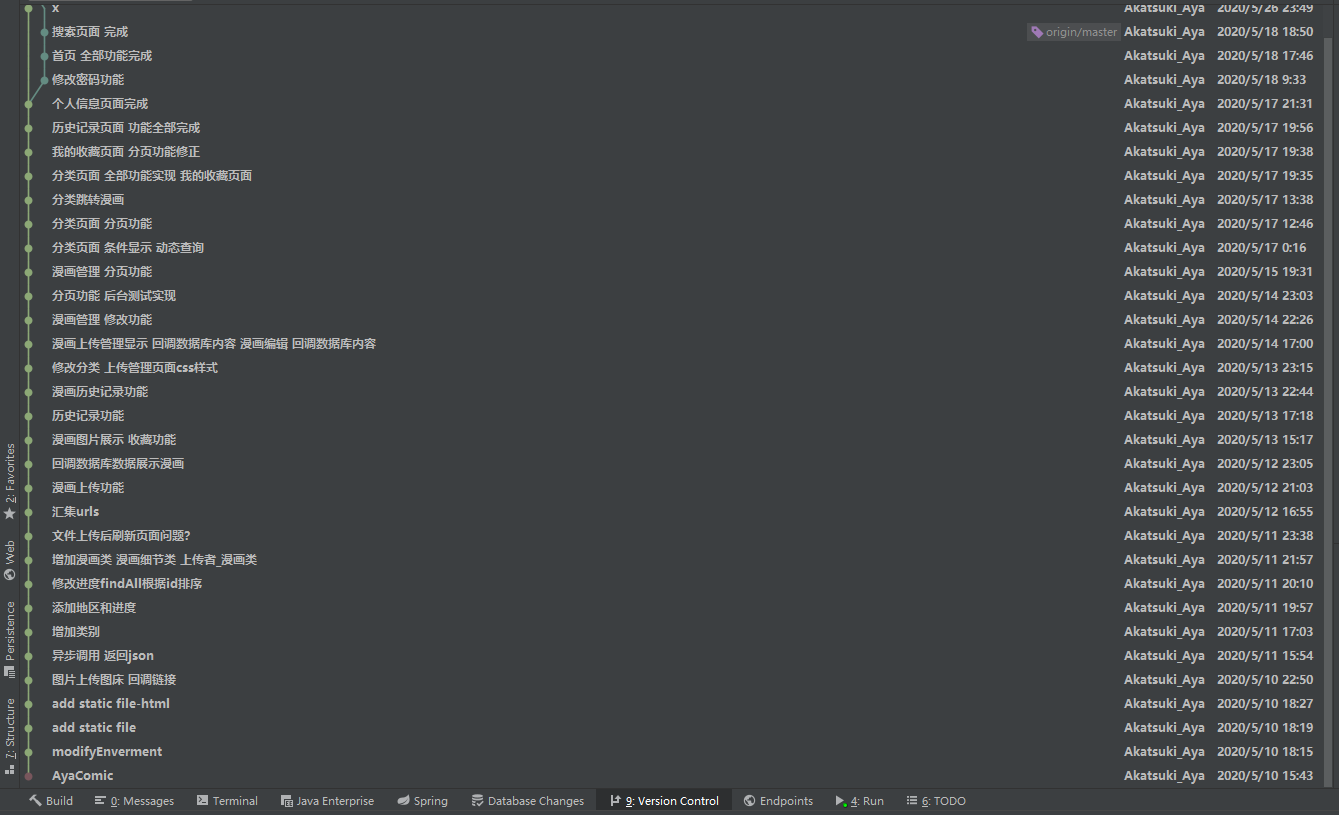


图 5.1 git提交历史

## 系统的完善

本系统可以完善的地方还是很多的。本身这个项目我也是有放在github上的，将来有条件的话可以继续对系统进行升级改造。首先就是想对原先需求功能的不足，像是评论系统，我还是特别希望加上去的。以及其他框架的追加，去优化改善系统的体验度。目前初步设想是加上Redis缓存，对数据库建立连接池来改善页面的加载速度。以及上文中所说的对系统细节的追加。还有想加上消息队列，当自己收藏的漫画更新时，用户可以收到通知。以及通过大数据分析，当用户点击浏览某一漫画时，可以显示出与它相关的其他漫画供用户选择。

## 总结与体会

总的来说，由于自身能力有限加上时间分配不均，导致系统做的不是十分的理想，只是实现系统作为基础的那部分功能。同时也通过此次毕设，积累了许多的经验，让我学习到了许多的知识，尤其是对于Springboot框架，有了一个较为基础的认识，让我了解到了这个框架的强大。其中在Springboot框架中蕴含的代码设计的哲理让我受益匪浅，比如依赖注入中工厂模式的设计思想，切向编程中代理模式的思想对我代码的编写上的影响很深。在这一次的毕设过程中我也意外了解到了需要其他的对我毕设影响很大的插件，像是Krajee和Bpage这两个插件，都是我在偶然的情况下才突然了解到的。但他们对我的影响是重大的，在本次的系统中有部分核心的功能都是依靠这两个插件才可以相对简单的实现的。让我确真的感受到了编程的历史果然是在前人的不断积累下成长的。前人展现了他们的互联网共享精神，制造了许多的“车轮”使我们少踩了许多的坑，我觉得我们也应该体会到他们的良苦用心，应该积极的去使用，避免重复造车形成一个正向的循环。我想这也是github设立的初衷吧。也正是因为这一点，让我觉得程序员是个时时刻刻都在不停创兴的职业，找到一个创意，通过代码实现，去渐渐的改变这个世界，优化我们的家园，改善我们的生活，这也是我所热爱这个职业的原因与初衷。以上虽然说，仅仅依靠这一次的毕设，可能依旧改变不了些许的现状，离自己的梦想也依旧遥远，但我相信此次经历中我所学到的知识，会在潜移默化中影响着我，使我成长，指引我走向理想的一块块基石。

致谢

首先要感谢的就是Idea这款软件。这绝对是我用过的最好的Java开发的IDE。并且给予学生免费使用，只要通过学校的邮箱进行申请就可以免费的使用这款软件。它内置的代码提示，代码自动生成大大提升了我的代码的编写效率。同时还要感谢阿里的代码约束插件，通过这一插件让我少踩了许多的坑，让我在潜意识里形成一个良好的代码编写风格。然后我要感谢github，由于在这次的开发过程中，我是在两个不同的开发环境上来回切换着进行开发的，一边是在公司的笔记本上，另一边则是家中的台式，通过github和git的提交，省去了我来回搬公司的笔记本回家的状况。只需要在最后提交下代码，回家后在下回来就行了，十分的方便，节省下了我许多的时间。然后我要感谢Krajee和Bpage两款插件的作者，是他们的共享精神，为我的分页功能和图片上传功能这两个核心功能点提供了不少的便利，减低我开发实现功能的难度。感谢许多博客的作者，是他们在我开发遇到困难，遇到解决不了的bug时，是他们给我提供了思路。最后感谢老师在我毕设过程中的督促和教导。

参考文献

[1] 中国网信网.中国互联网络发展状况统计报告[EB/OL].http://www.cac.gov.cn/2019-08/30/c\_1124939590.htm,2019-8-30.

[2] Bootstrap中文网.Bootstrap中文网[EB/OL].https://www.bootcss.com/,2020-01-16.

[3] 杨冠宝,高海慧.码出高效Java开发手册[M].电子工业出版社:北京,2018-10

[4] 赵国玲,王宏,柴大鹏.java语言程序设计(第2版), 机械工业出版社, 2010.03

[5] 王福强 SpringBoot揭秘：快速构建微服务体系, 机械工业出版社, 2016.05

[6] Den Odell(著) 邝健威，厉海洋(译)， 精通javascript开发， 人民邮电出版社，2015.09

[7] 李晓斌 ,Div+CSS 3+jQuery网页布局案例精粹 , 电子工业出版社，2015 .06

[8] GitHub.Krajee开源插件[EB/OL].https://github.com/kartik-v/yii2-krajee-base,2019-5-29.

[9] Bpage开源插件[EB/OL].https://terryz.github.io/bpage/index.html,2016-6-8.

[10] John Carnell.Spring Microservices in Action;Spring Microservices in Action teaches you how to build microservice-based applications using Java and the Spring platform.[M].Manning Publications:Lee Fitzpatrick,2017:127-144.

[11] Greg L. Turnquist.Learning Spring Boot 2.0;Simplify the development of lightning fast applications based on microservices and reactive programming[M].Packt Publishing:ebooks Account,2017-12-6:44-58.