附录A 软件需求分析报告文档模板

[1 综合描述-------------------------------------------------------------------------------2](#_Toc406961458)

[1.1 产品的功能-------------------------------------------------------------------------2](#_Toc406961459)

[1.2 用户类和特性----------------------------------------------------------------------2](#_Toc406961460)

[1.3 运行环境----------------------------------------------------------------------------2](#_Toc406961461)

[1.4 设计和实现上的限制-------------------------------------------------------------2](#_Toc406961462)

[2 外部接口需求-------------------------------------------------------------------------3](#_Toc406961463)

[2.1 用户界面----------------------------------------------------------------------------3](#_Toc406961464)

[2.2 硬件接口----------------------------------------------------------------------------3](#_Toc406961465)

[2.3 软件接口----------------------------------------------------------------------------3](#_Toc406961466)

[2.4 通讯接口----------------------------------------------------------------------------3](#_Toc406961467)

[3 系统功能需求-------------------------------------------------------------------------3](#_Toc406961468)

[4 其它非功能需求----------------------------------------------------------------------4](#_Toc406961469)

[4.1 性能需求----------------------------------------------------------------------------4](#_Toc406961470)

[4.2 安全措施需求----------------------------------------------------------------------4](#_Toc406961471)

[4.3 安全性需求-------------------------------------------------------------------------4](#_Toc406961472)

[4.4 软件质量属性----------------------------------------------------------------------4](#_Toc406961473)

[4.5 业务规则----------------------------------------------------------------------------5](#_Toc406961474)

# 

**# 看点咨询后台管理系统软件需求分析报告文档**

**## 1 综合描述**

本节概述了看点咨询后台管理系统的作用范围、运行环境、用户类和特性、设计和实现上的限制。

**### 1.1 产品的功能**

看点咨询后台管理系统是一个为看点咨询平台提供内容管理、用户管理、日志管理、评论管理等功能的系统。本系统的主要功能如下：

- 用户管理：可以新增、删除用户，以及查看用户的基本信息和活动记录。

- 栏目管理：可以添加、删除、修改栏目，以及设置栏目的排序、显示、隐藏等属性。

- 资讯模块：可以发布、删除、修改资讯，以及设置资讯的标题、摘要、正文、图片、视频、标签、分类、置顶、推荐等属性。

- 日志管理：可以查看系统的操作日志、异常日志、访问日志等，以及按照时间、类型、用户、IP等条件进行筛选和搜索。

- 评论管理：可以查看、删除、屏蔽用户对资讯的评论，以及按照时间、内容、用户、资讯等条件进行筛选和搜索。

**### 1.2 用户类和特性**

本系统的用户分为两类：管理员和普通用户。管理员是指有权限登录后台管理系统的用户，普通用户是指在看点咨询平台上浏览、评论资讯的用户。它们的相关特征如下：

- 管理员：管理员可以使用本系统的所有功能，包括用户管理、栏目管理、资讯模块、日志管理、评论管理等。管理员需要通过用户名和密码进行身份验证，以及通过验证码进行安全验证。管理员可以修改自己的密码和个人信息，也可以查看其他管理员的信息和操作记录。

- 普通用户：普通用户可以在看点咨询平台上查看、搜索、收藏、分享、评论资讯，以及查看、修改自己的个人信息和评论记录。普通用户需要通过手机号和验证码进行注册和登录，也可以使用第三方账号进行绑定和登录。普通用户的信息和行为会被本系统记录和管理。

**### 1.3 运行环境**

本系统的运行环境如下：

- 硬件平台：服务器端采用M1Pro处理器，32GB内存，1TB硬盘，千兆网卡；客户端采用PC或移动设备，具有网络连接功能。

- 操作系统和版本：服务器端采用Ubuntu操作系统，客户端采用Windows 10、操作系统。

- 支撑环境和版本：服务器端采用JDK等支撑环境，客户端采用Chrome 87、Firefox 84、Safari 14等浏览器。

- 其它与该软件有关的软件组件：服务器端采用MySql等第三方库，客户端采用IDEA等第三方软件。

- 与该软件共存的应用程序：服务器端与看点咨询平台的前端应用程序共存，客户端与看点咨询平台的移动应用程序共存。

**### 1.4 设计和实现上的限制**

本系统的设计和实现上的限制如下：

- 必须使用的特定技术、工具、编程语言和数据库：服务器端必须使用JDK，MySql；客户端必须使用HTML、CSS、JavaScript语言、Visual Studio Code工具、Vue.js框架，IDEA。

- 要求遵循的开发规范和标准：服务器端和客户端都要遵循Microsoft的编码规范和风格指南，以及看点咨询平台的开发规范和标准。

**## 2 外部接口需求**

本节描述了本系统与外部组件的连接需求，包括用户界面、硬件接口、软件接口、通讯接口等。

**### 2.1 用户界面**

本系统的用户界面分为两部分：后台管理系统的用户界面和看点咨询平台的用户界面。后台管理系统的用户界面是指管理员在浏览器中访问的页面，看点咨询平台的用户界面是指普通用户在移动设备上访问的页面。它们的逻辑特征如下：

- 后台管理系统的用户界面：后台管理系统的用户界面采用响应式布局，适应不同的屏幕尺寸和分辨率。用户界面的主要元素包括导航栏、侧边栏、内容区域、页脚等。导航栏显示系统的logo、管理员的姓名和头像、退出按钮等；侧边栏显示系统的功能菜单，可以展开和收起；内容区域显示各个功能的操作界面，包括表格、表单、图表、弹窗等；页脚显示系统的版权信息和联系方式等。用户界面的风格和色彩与看点咨询平台的前端应用程序保持一致，简洁、清晰、美观。

- 看点咨询平台的用户界面：看点咨询平台的用户界面采用原生或混合开发，适应不同的移动设备和操作系统。用户界面的主要元素包括顶部栏、底部栏、内容区域等。顶部栏显示系统的logo、搜索框、消息提醒等；底部栏显示系统的主要功能按钮，包括首页、发现、我的等；内容区域显示各个功能的操作界面，包括资讯列表、资讯详情、评论列表、评论详情、个人信息、收藏列表、设置等。用户界面的风格和色彩与看点咨询平台的网页应用程序保持一致，简洁、清晰、美观。

**### 2.2 硬件接口**

本系统与硬件接口的特征如下：

- 服务器端与硬件接口的特征：服务器端与服务器的硬件接口主要包括CPU、内存、硬盘、网卡等。服务器端需要通过硬件接口进行数据的处理、存储、传输等操作。服务器端需要保证硬件接口的稳定性、可靠性、安全性、性能等。

- 客户端与硬件接口的特征：客户端与客户端设备的硬件接口主要包括屏幕、键盘、鼠标、触摸屏、摄像头、麦克风、扬声器等。客户端需要通过硬件接口进行数据的输入、输出、展示等操作。客户端需要保证硬件

接口的兼容性、适配性、易用性等。

**### 2.3 软件接口**

本系统与软件接口的特征如下：

- 服务器端与软件接口的特征：服务器端与.NET Framework、SQL Server、IIS等软件接口主要用于提供开发、运行、部署、管理等功能。服务器端需要通过软件接口进行数据的操作、访问、交换等操作。服务器端需要保证软件接口的版本、配置、协议、格式等。

- 客户端与软件接口的特征：客户端与jQuery、Bootstrap、Vue.js等软件接口主要用于提供前端开发、交互、渲染等功能。客户端需要通过软件接口进行数据的展示、处理、传输等操作。客户端需要保证软件接口的兼容性、稳定性、性能等。

**### 2.4 通讯接口**

本系统与通讯接口的特征如下：

- 服务器端与通讯接口的特征：服务器端与客户端之间的通讯接口主要采用HTTP协议，以及JSON格式的数据交换。服务器端需要通过通讯接口进行数据的响应、发送、接收等操作。服务器端需要保证通讯接口的安全性、可靠性、效率等。

- 客户端与通讯接口的特征：客户端与服务器端之间的通讯接口主要采用HTTP协议，以及JSON格式的数据交换。客户端需要通过通讯接口进行数据的请求、发送、接收等操作。客户端需要保证通讯接口的可用性、正确性、及时性等。

**## 3 系统功能需求**

本节详细描述了本系统的功能需求，按照使用实例的方式进行组织，每个使用实例对应一个系统功能。每个系统功能需求都有唯一的标识，以及详细的描述和规格说明。

**### 3.1 用户管理**

本系统功能需求的标识为：FR-UM-01

本系统功能需求的描述为：管理员可以通过本系统对用户进行管理，包括新增、删除用户，以及查看用户的基本信息和活动记录。

本系统功能需求的规格说明为：

- 前置条件：管理员已经登录后台管理系统，并且有用户管理的权限。

- 后置条件：管理员对用户的操作成功执行，并且系统记录相应的日志。

- 基本流程：

- 管理员在侧边栏点击“用户管理”菜单，进入用户管理界面。

- 管理员在用户管理界面可以看到用户的列表，每个用户的信息包括用户ID、手机号、昵称、头像、注册时间、最后登录时间、状态等。

- 管理员可以在用户管理界面的顶部输入用户的手机号或昵称进行搜索，也可以按照注册时间或最后登录时间进行排序。

- 管理员可以在用户管理界面的底部点击“新增用户”按钮，弹出新增用户的表单，输入用户的手机号、昵称、头像、密码等信息，点击“确定”按钮，系统验证用户信息的合法性，如果合法，系统创建新的用户，并返回用户管理界面，显示新的用户列表，如果不合法，系统提示错误信息，让管理员重新输入。

- 管理员可以在用户管理界面的列表中点击某个用户的“删除”按钮，弹出删除用户的确认框，显示用户的手机号和昵称，询问管理员是否确定删除该用户，如果管理员点击“确定”按钮，系统删除该用户，并返回用户管理界面，显示新的用户列表，如果管理员点击“取消”按钮，系统取消删除操作，返回用户管理界面，不做任何改变。

- 管理员可以在用户管理界面的列表中点击某个用户的“详情”按钮，进入用户详情界面，显示用户的基本信息和活动记录。用户的基本信息包括用户ID、手机号、昵称、头像、注册时间、最后登录时间、状态等，用户的活动记录包括用户浏览、收藏、分享、评论的资讯的列表，每个资讯的信息包括资讯ID、标题、摘要、图片、时间等。

- 管理员可以在用户详情界面的顶部点击“返回”按钮，返回用户管理界面，不做任何改变。

- 备选流程：

- 如果管理员在新增用户的表单中输入的手机号已经存在，系统提示“该手机号已经被注册，请更换其他手机号”。

- 如果管理员在新增用户的表单中输入的昵称已经存在，系统提示“该昵称已经被使用，请更换其他昵称”。

- 如果管理员在新增用户的表单中输入的密码不符合安全规则，系统提示“密码必须包含大小写字母、数字、特殊符号，且长度不少于8位”。

- 如果管理员在删除用户的确认框中点击“确定”按钮，系统删除该用户时发生异常，系统提示“删除用户失败，请稍后重试”。

- 如果管理员在用户管理界面或用户详情界面的任何操作时发生网络中断或超时，系统提示“网络异常，请检查网络连接”。

**### 3.2 资讯模块**

本系统功能需求的描述为：管理员可以通过本系统对资讯进行管理，包括发布、删除、修改资讯，以及设置资讯的标题、摘要、正文、图片、视频、标签、分类、置顶、推荐等属性。

本系统功能需求的规格说明为：

- 前置条件：管理员已经登录后台管理系统，并且有资讯模块的权限。

- 后置条件：管理员对资讯的操作成功执行，并且系统记录相应的日志。

- 基本流程：

- 管理员在侧边栏点击“资讯模块”菜单，进入资讯模块界面。

- 管理员在资讯模块界面可以看到资讯的列表，每个资讯的信息包括资讯ID、标题、摘要、图片、时间、状态、分类、标签、置顶、推荐等。

- 管理员可以在资讯模块界面的顶部输入资讯的标题或摘要进行搜索，也可以按照时间、状态、分类、标签、置顶、推荐等条件进行筛选和排序。

- 管理员可以在资讯模块界面的底部点击“发布资讯”按钮，弹出发布资讯的表单，输入资讯的标题、摘要、正文、图片、视频、标签、分类、置顶、推荐等信息，点击“确定”按钮，系统验证资讯信息的合法性，如果合法，系统创建新的资讯，并返回资讯模块界面，显示新的资讯列表，如果不合法，系统提示错误信息，让管理员重新输入。

- 管理员可以在资讯模块界面的列表中点击某个资讯的“删除”按钮，弹出删除资讯的确认框，显示资讯的标题，询问管理员是否确定删除该资讯，如果管理员点击“确定”按钮，系统删除该资讯，并返回资讯模块界面，显示新的资讯列表，如果管理员点击“取消”按钮，系统取消删除操作，返回资讯模块界面，不做任何改变。

- 管理员可以在资讯模块界面的列表中点击某个资讯的“修改”按钮，弹出修改资讯的表单，显示资讯的标题、摘要、正文、图片、视频、标签、分类、置顶、推荐等信息，让管理员进行修改，点击“确定”按钮，系统验证资讯信息的合法性，如果合法，系统更新该资讯，并返回资讯模块界面，显示新的资讯列表，如果不合法，系统提示错误信息，让管理员重新输入。

- 备选流程：

- 如果管理员在发布资讯的表单中输入的资讯标题已经存在，系统提示“该资讯标题已经被使用，请更换其他资讯标题”。

- 如果管理员在发布资讯的表单中输入的资讯摘要超过200字，系统提示“资讯摘要不能超过200字”。

- 如果管理员在发布资讯的表单中输入的资讯图片或视频的格式或大小不符合要求，系统提示“资讯图片或视频的格式或大小不符合要求，请重新选择”。

- 如果管理员在删除资讯的确认框中点击“确定”按钮，系统删除该资讯时发生异常，系统提示“删除资讯失败，请稍后重试”。

- 如果管理员在修改资讯的表单中输入的资讯标题已经存在，系统提示“该资讯标题已经被使用，请更换其他资讯标题”。

- 如果管理员在修改资讯的表单中输入的资讯摘要超过200字，系统提示“资讯摘要不能超过200字”。

- 如果管理员在修改资讯的表单中输入的资讯图片或视频的格式或大小不符合要求，系统提示“资讯图片或视频的格式或大小不符合要求，请重新选择”。

- 如果管理员在资讯模块界面或资讯详情界面的任何操作时发生网络中断或超时，系统提示“网络异常，请检查网络连接”。

4. 其它非功能需求

4.1 性能需求

- 系统应能够支持至少1000个并发用户访问，响应时间不超过3秒。

- 系统应能够保证每天处理至少10万条资讯数据，存储空间不少于1TB。

- 系统应能够实时更新资讯内容，保证数据的一致性和准确性。

- 系统应能够提供日志管理功能，记录所有用户的操作和异常事件，方便追溯和分析。

4.2 安全措施需求

- 系统应采用HTTPS协议进行数据传输，防止数据被窃取或篡改。

- 系统应采用加密算法对用户的密码和敏感数据进行加密存储，防止数据泄露。

- 系统应提供备份和恢复功能，定期备份数据，防止数据丢失或损坏。

- 系统应提供防火墙和反病毒软件，防止系统遭受网络攻击或病毒感染。

4.3 安全性需求

- 系统应提供用户管理功能，对不同角色的用户进行分级授权，限制用户的访问权限和操作范围。

- 系统应提供身份认证功能，要求用户输入用户名和密码进行登录，防止未授权的用户进入系统。

- 系统应提供修改密码功能，要求用户在第一次登录后，必须更改他的系统预置登录密码，系统预置的登录密码不能重用。

- 系统应提供注销功能，要求用户在离开系统时，必须注销登录，防止他人利用用户的账号进行非法操作。

4.4 软件质量属性

- 系统应具有良好的易用性，提供友好的用户界面，清晰的操作指引，简洁的交互流程，方便用户使用。

- 系统应具有良好的可维护性，遵循统一的编码规范，使用模块化的设计，提供详细的注释和文档，方便开发人员维护和修改。

- 系统应具有良好的可扩展性，使用开放的技术和标准，提供接口和插件，方便系统与其他系统集成和扩展。

- 系统应具有良好的可测试性，使用自动化的测试工具，提供测试用例和测试数据，方便测试人员测试和验证。

4.5 业务规则

- 用户管理

- 系统管理员可以新增，删除，修改用户的信息和角色。

- 系统管理员可以查看用户的登录日志和操作日志。

- 用户可以修改自己的密码和个人信息。

- 栏目管理

- 系统管理员可以新增，删除，修改栏目的名称和描述。

- 系统管理员可以为栏目分配资讯编辑和审核员。

- 资讯编辑可以为栏目添加，删除，修改资讯内容。

- 资讯审核员可以为栏目审核，发布，撤回资讯内容。

- 资讯模块

- 资讯编辑可以为资讯内容添加，删除，修改标题，正文，图片，视频，链接等元素。

- 资讯编辑可以为资讯内容设置发布时间，置顶，推荐等属性。

- 资讯编辑可以提交资讯内容给资讯审核员进行审核。

- 资讯审核员可以查看，修改，通过，拒绝资讯内容的审核请求。

- 资讯审核员可以发布，撤回已通过审核的资讯内容。

- 日志管理

- 系统管理员可以查看，导出，删除系统的登录日志和操作日志。

- 系统管理员可以根据时间，用户，操作类型等条件进行日志查询和筛选。

- 评论管理

- 系统管理员可以查看，删除，屏蔽用户对资讯内容的评论。

- 系统管理员可以根据时间，用户，内容，状态等条件进行评论查询和筛选。

附录B 软件概要设计报告文档模板

[1. 设计概述-------------------------------------------------------------------------------1](#_Toc406963852)

[2. 系统逻辑设计-------------------------------------------------------------------------2](#_Toc406963853)

[3. 数据库设计----------------------------------------------------------------------------2](#_Toc406963854)

1. 设计概述

本软件产品是一个看点咨询后台管理系统，主要用于管理用户，栏目，资讯，日志，评论等功能。本系统的目标是提供一个高效，安全，易用的平台，让管理员可以方便地发布，审核，管理各种资讯内容，提高用户的阅读体验和满意度。

进行概要设计时，应遵循以下设计原则和方法：

- 采用模块化的设计思想，将系统划分为多个功能模块，降低系统的复杂度，提高系统的可维护性和可扩展性。

- 采用面向对象的设计方法，使用类图，时序图，状态图等工具，描述系统的对象，属性，方法，关系，行为，状态等方面，提高系统的逻辑性和一致性。

- 采用MVC的设计模式，将系统分为模型，视图，控制器三层，实现系统的分层，分离，分工，提高系统的可重用性和可测试性。

2. 系统逻辑设计

本节根据软件产品需求规格说明书和软件产品数据字典，建立系统的逻辑模型，给出系统结构图，描述系统的功能模块和子系统之间的关系。

系统结构图如下：

!系统结构图

系统结构图说明：

- 系统由五个功能模块组成，分别是用户管理，栏目管理，资讯模块，日志管理，评论管理。

- 用户管理模块负责管理系统的用户信息和角色，提供用户的新增，删除，修改，查询，登录，注销等功能。

- 栏目管理模块负责管理系统的栏目信息，提供栏目的新增，删除，修改，查询，分配等功能。

- 资讯模块负责管理系统的资讯内容，提供资讯的添加，删除，修改，查询，审核，发布，撤回等功能。

- 日志管理模块负责管理系统的日志信息，提供日志的查看，导出，删除，查询，筛选等功能。

- 评论管理模块负责管理系统的评论信息，提供评论的查看，删除，屏蔽，查询，筛选等功能。

- 系统划分为两个子系统，分别是后台管理子系统和前台展示子系统。

- 后台管理子系统包含用户管理，栏目管理，资讯模块，日志管理，评论管理五个功能模块，主要为系统管理员，资讯编辑，资讯审核员提供服务，实现系统的管理和维护功能。

- 前台展示子系统主要为普通用户提供服务，实现系统的展示和交互功能，让用户可以浏览，阅读，评论，分享各种资讯内容。

3. 数据库设计

本软件产品需要使用MySQL数据库来存储和管理系统的数据，包括用户，栏目，资讯，日志，评论等表。数据库设计的详细内容参见软件产品数据库设计说明文档。

附录C 软件详细设计报告文档模板

[1. 界面设计--------------------------------------------------------------------------------2](#_Toc406964899)

[2. 过程设计--------------------------------------------------------------------------------9](#_Toc406964912)

1. 界面设计

本节描述了本系统的人机界面设计，包括后台管理子系统和前台展示子系统的界面设计。本系统的界面设计遵循以下原则：

- 界面简洁，美观，符合用户的使用习惯和审美标准。

- 界面布局合理，突出重点，突出功能，方便用户操作和浏览。

- 界面风格统一，色彩搭配协调，字体大小适中，图标清晰，提示信息明确。

- 界面交互友好，提供必要的帮助和反馈，处理异常和错误，防止误操作。

后台管理子系统的界面设计如下：

- 登录界面：该界面用于用户输入用户名和密码进行登录，登录成功后进入主界面，登录失败后给出提示信息。该界面还提供修改密码和忘记密码的功能，以及系统的logo和名称。

- 主界面：该界面用于显示系统的主要功能模块，包括用户管理，栏目管理，资讯模块，日志管理，评论管理。用户可以通过点击相应的模块进入子界面，或者通过下拉菜单选择具体的功能。该界面还提供用户的个人信息，系统的时间，注销和退出的功能，以及系统的logo和名称。

- 用户管理界面：该界面用于显示和管理系统的用户信息和角色，提供用户的新增，删除，修改，查询，登录日志，操作日志等功能。该界面采用表格的形式展示用户的列表，每个用户的信息包括用户名，姓名，角色，状态，最近登录时间，操作等。用户可以通过输入用户名或姓名进行查询，或者通过点击表头进行排序。用户可以通过点击操作列的按钮进行用户的删除，修改，登录日志，操作日志等功能。用户可以通过点击新增按钮进行用户的新增功能。该界面还提供分页，全选，反选，批量删除等功能。

- 栏目管理界面：该界面用于显示和管理系统的栏目信息，提供栏目的新增，删除，修改，查询，分配等功能。该界面采用树形的形式展示栏目的层级结构，每个栏目的信息包括栏目名称，栏目描述，资讯编辑，资讯审核员，操作等。用户可以通过输入栏目名称进行查询，或者通过拖拽栏目进行移动。用户可以通过点击操作列的按钮进行栏目的删除，修改，分配等功能。用户可以通过点击新增按钮进行栏目的新增功能。该界面还提供刷新，展开，收起等功能。

- 资讯模块界面：该界面用于显示和管理系统的资讯内容，提供资讯的添加，删除，修改，查询，审核，发布，撤回等功能。该界面采用表格的形式展示资讯的列表，每个资讯的信息包括资讯标题，资讯栏目，资讯编辑，资讯状态，发布时间，操作等。用户可以通过输入资讯标题或栏目名称进行查询，或者通过点击表头进行排序。用户可以通过点击操作列的按钮进行资讯的删除，修改，审核，发布，撤回等功能。用户可以通过点击添加按钮进行资讯的添加功能。该界面还提供分页，全选，反选，批量删除等功能。

- 日志管理界面：该界面用于显示和管理系统的日志信息，提供日志的查看，导出，删除，查询，筛选等功能。该界面采用表格的形式展示日志的列表，每个日志的信息包括日志类型，日志内容，操作用户，操作时间，操作等。用户可以通过输入日志类型，日志内容，操作用户，操作时间等条件进行查询和筛选，或者通过点击表头进行排序。用户可以通过点击操作列的按钮进行日志的查看，导出，删除等功能。该界面还提供分页，全选，反选，批量删除等功能。

- 评论管理界面：该界面用于显示和管理系统的评论信息，提供评论的查看，删除，屏蔽，查询，筛选等功能。该界面采用表格的形式展示评论的列表，每个评论的信息包括评论内容，评论用户，评论资讯，评论时间，评论状态，操作等。用户可以通过输入评论内容，评论用户，评论资讯，评论时间等条件进行查询和筛选，或者通过点击表头进行排序。用户可以通过点击操作列的按钮进行评论的查看，删除，屏蔽等功能。该界面还提供分页，全选，反选，批量删除等功能。

2.1 用户管理模块

2.1.1 用户新增流程

描述：用户新增流程主要包括管理员输入新增用户信息、系统验证信息合法性、生成用户ID、将用户信息写入数据库等步骤。

2.1.2 用户删除流程

描述：用户删除流程包括管理员输入要删除的用户ID、系统验证管理员权限、从数据库中删除用户信息等步骤。

2.2 栏目管理模块

2.2.1 栏目添加流程

描述：栏目添加流程包括管理员输入新栏目信息、系统验证信息合法性、生成栏目ID、将栏目信息写入数据库等步骤。

2.2.2 栏目删除流程

描述：栏目删除流程包括管理员输入要删除的栏目ID、系统验证管理员权限、从数据库中删除栏目信息等步骤。

2.3 资讯管理模块

2.3.1 资讯发布流程

描述：资讯发布流程包括编辑人员输入资讯内容、选择所属栏目、系统验证信息合法性、生成资讯ID、将资讯信息写入数据库等步骤。

2.3.2 资讯删除流程

描述：资讯删除流程包括编辑人员输入要删除的资讯ID、系统验证编辑人员权限、从数据库中删除资讯信息等步骤。

2.3.3 资讯修改流程

描述：资讯修改流程包括编辑人员输入要修改的资讯ID、编辑新的资讯内容、系统验证编辑人员权限、更新数据库中的资讯信息等步骤。

2.4 评论管理模块

2.4.1 评论查看流程

描述：评论查看流程包括编辑人员选择要查看的资讯、系统验证编辑人员权限、从数据库中获取评论信息等步骤。

2.4.2 评论删除流程

描述：评论删除流程包括编辑人员输入要删除的评论ID、系统验证编辑人员权限、从数据库中删除评论信息等步骤。

附录D 软件实现报告文档模板

[1. 程序设计语言的选择 2](#_Toc406966548)

[2. 编码风格说明 2](#_Toc406966549)

[3. 详细实现过程及代码 2](#_Toc406966550)

1. **程序设计语言的选择** 1.1 **选择背景**
   * 本系统选择Java Spring Boot作为主要的程序设计语言，主要出于其在企业级应用开发中的广泛应用和成熟的生态系统。

1.2 **横向比较**

* + 与其他语言相比，Java具有跨平台性、强大的社区支持、良好的性能和广泛的库支持等优势。

1.3 **纵向比较**

* + 在与其他Java框架的比较中，Spring Boot作为一个轻量级、开发效率高的框架，更适合本系统的需求。

1.4 **选择理由**

* + Java Spring Boot提供了全面的开发工具和库，对于后台管理系统而言，具有良好的可维护性、可扩展性和高度的开发效率。

1. **编码风格说明** 2.1 **注释**
   * 本系统采用Javadoc风格的注释，注重对方法、类、接口的说明，以及关键算法的注释，提高代码的可读性。

2.2 **数据说明**

* + 所有关键数据结构和变量都将在代码中进行明确的注释说明，包括其作用、取值范围等。

2.3 **命名规范**

* + 遵循Java的命名规范，类名使用驼峰命名法，方法和变量名使用小驼峰命名法，常量使用全大写下划线分隔。

2.4 **代码结构**

* + 代码将按照模块进行划分，每个模块包含单一的职责，降低耦合度，提高代码的可维护性。

1. **详细实现过程及代码**

Application.yml

# Web服务器配置

server:

# 端口号

port: 8989

# Spring相关配置

spring:

# 数据源配置

datasource:

# 数据库连接URL（注意：需按照自己的实际情况填写ip、端口及数据库名）

url: jdbc:mysql://localhost:3306/briup\_cms?useSSL=false&useUnicode=true&characterEncoding=utf8&serverTimeZone=Asia/Shanghai

# 数据库连接账号

username: root

# 数据库连接密码

password: 123123123

# MyBatis相关配置

mybatis:

# 映射文件所在路径

mapper-locations: classpath:mappers/\*Mapper.xml

configuration:

# 开启MyBatis执行日志

log-impl: org.apache.ibatis.logging.stdout.StdOutImpl

# MyBatisPlus相关配置

mybatis-plus:

global-config:

db-config:

# 逻辑删除中表示删除状态的字段值（默认为1）

logic-delete-value: 1

# 逻辑删除中表示未删除状态的字段值（默认为0）

logic-not-delete-value: 0

# 定义表示逻辑删除状态的字段名（可以通过TableLogic注解声明）

logic-delete-field: deleted

configuration:

# 开启下划线命名和驼峰命名之间的自动映射转换功能

map-underscore-to-camel-case: true

# 日志相关配置

logging:

# 日志输出级别

level:

root: info

# 自定义配置

briup:

config:

## OSS七牛云存储配置 ##

# 桶名称（存储空间名称）

oss-bucket: group-sxau-yy

# 公钥

oss-access-key: NbYOK7d5sN3Zq5WqF-wkTGHmwPBQLbASdAuhcREF

# 私钥

oss-secret-key: ZQhcN15rXw4j-zgbb9\_Af1pL9MWYNpw7zZInF8YW

# URL固定前缀

oss-base-url: http://s75evsxpq.hb-bkt.clouddn.com/

# 默认文件名称

oss-default-file-name: noname.jpg

## 加密相关配置 ##

# 加密强度

security-strength: 6

# 干扰因子（使用随机UUID生成）

security-secret: ${random.uuid}

## 跨域配置 ##

# 允许跨域访问的资源

cors-mapping: "/\*\*"

# 允许跨域访问的客户端源

cors-allowed-origins: "\*"

# 允许跨域访问使用的HTTP请求方法

cors-allowed-methods:

- "GET"

- "POST"

- "PUT"

- "DELETE"

# 允许跨域访问携带的HTTP请求头部

cors-allowed-headers: "\*"

# 是否允许携带Cookie

cors-allow-credentials: false

# 预检请求缓存有效时长（单位：秒）

cors-max-age: 3600

## 访问权限配置 ##

# 需要拦截的请求URL

auth-include-path-patterns: /\*\*

# 不需要拦截的请求URL

auth-exclude-path-patterns:

- /login

- /logout

- /upload

- /auth/category/\*\*

- /auth/log/\*\*

- /slideshow/\*\*

## 导出、下载相关配置 ##

# 导出Excel使用的编码格式

excel-export-encoding: UTF-8

# 导出Excel的文件类型（Office2003之前对应XLS，Office2003之后对应XLSX）

excel-export-file-type: XLSX

# 导出Excel时使用的Content-Type头部字段值

excel-export-content-type: application/octet-stream;charset=${briup.config.excel-export-encoding}

# 导出Excel时使用的Content-Disposition头部字段值

excel-export-content-disposition: attachment;filename=%s

## 其他配置 ##

# 默认时间格式（解析请求参数时使用）

default-date-pattern: yyyy-MM-dd HH:mm:ss

# IP查询服务地址

ip-query-server-host: http://whois.pconline.com.cn/ipJson.jsp

# pom.xml

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.briup</groupId>

<artifactId>briup-cms</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>briup-cms</name>

<description>Demo project for Spring Boot</description>

<properties>

<java.version>1.8</java.version>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>

<spring-boot.version>2.7.17</spring-boot.version>

<mybatis-version>2.2.2</mybatis-version>

<jjwt.version>0.9.1</jjwt.version>

<qiniu.version>[7.13.0, 7.13.99]</qiniu.version>

<mybatis-plus.version>3.5.2</mybatis-plus.version>

</properties>

<dependencies>

<!-- SpringWeb启动器 -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<!-- SpringAOP启动器 -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-aop</artifactId>

</dependency>

<!-- SpringAOP启动器 -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-security</artifactId>

</dependency>

<!-- 为自定义配置生成元信息 -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-configuration-processor</artifactId>

</dependency>

<!-- MyBatis启动器 -->

<dependency>

<groupId>org.mybatis.spring.boot</groupId>

<artifactId>mybatis-spring-boot-starter</artifactId>

<version>${mybatis-version}</version>

</dependency>

<!-- MyBatisPlus启动器 -->

<dependency>

<groupId>com.baomidou</groupId>

<artifactId>mybatis-plus-boot-starter</artifactId>

<version>${mybatis-plus.version}</version>

</dependency>

<!-- MySQL驱动 -->

<dependency>

<groupId>com.mysql</groupId>

<artifactId>mysql-connector-j</artifactId>

<scope>runtime</scope>

</dependency>

<!-- Lombok开发工具 -->

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

<optional>true</optional>

</dependency>

<!-- SpringBoot测试框架启动器 -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

<!-- JWT依赖 -->

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt</artifactId>

<version>${jjwt.version}</version>

</dependency>

<!-- &lt;!&ndash; 七牛云SDK依赖 &ndash;&gt;-->

<!-- <dependency>-->

<!-- <groupId>com.qiniu</groupId>-->

<!-- <artifactId>qiniu-java-sdk</artifactId>-->

<!-- <version>${qiniu.version}</version>-->

<!-- </dependency>-->

<!-- 七牛云SDK依赖 -->

<dependency>

<groupId>com.qiniu</groupId>

<artifactId>qiniu-java-sdk</artifactId>

<version>[7.7.0, 7.7.99]</version>

</dependency>

<!-- <dependency>-->

<!-- <groupId>org.apache.poi</groupId>-->

<!-- <artifactId>poi-ooxml</artifactId>-->

<!-- <version>3.9</version>-->

<!-- </dependency>-->

<!--Alibaba EasyExcel依赖 -->

<dependency>

<groupId>com.alibaba</groupId>

<artifactId>easyexcel</artifactId>

<version>3.3.2</version>

</dependency>

<dependency>

<groupId>com.alibaba.fastjson2</groupId>

<artifactId>fastjson2</artifactId>

<version>2.0.37</version>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-redis</artifactId>

<version>1.4.7.RELEASE</version>

</dependency>

</dependencies>

<dependencyManagement>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-dependencies</artifactId>

<version>${spring-boot.version}</version>

<type>pom</type>

<scope>import</scope>

</dependency>

</dependencies>

</dependencyManagement>

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.8.1</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

<encoding>UTF-8</encoding>

</configuration>

</plugin>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

<version>${spring-boot.version}</version>

<configuration>

<mainClass>com.briup.cms.BriupCmsApplication</mainClass>

<!--<skip>true</skip>-->

</configuration>

<executions>

<execution>

<id>repackage</id>

<goals>

<goal>repackage</goal>

</goals>

</execution>

</executions>

</plugin>

</plugins>

</build>

</project>

UserServiceImpl.java

package com.briup.cms.service.impl;  
  
import com.baomidou.mybatisplus.core.conditions.query.LambdaQueryWrapper;  
import com.baomidou.mybatisplus.core.metadata.IPage;  
import com.briup.cms.common.exception.CmsException;  
import com.briup.cms.common.model.entity.User;  
import com.briup.cms.common.model.ext.UserExt;  
import com.briup.cms.common.util.PageUtil;  
import com.briup.cms.common.util.ResultCode;  
import com.briup.cms.common.util.SecurityUtil;  
import com.briup.cms.dao.UserMapper;  
import com.briup.cms.service.UserService;  
import lombok.RequiredArgsConstructor;  
import org.springframework.stereotype.Service;  
import org.springframework.util.DigestUtils;  
import org.springframework.util.StringUtils;  
  
import java.util.Date;  
import java.util.List;  
  
*/\*\*  
 \* @author YuYan  
 \* @date 2024-01-09 19:42:40  
 \*/*@Service  
@RequiredArgsConstructor  
public class UserServiceImpl implements UserService {  
  
 *//注入用户模块Dao层对象* private final UserMapper userMapper;  
  
 *//注入安全加密工具类对象* private final SecurityUtil securityUtil;  
  
 *//新增用户信息* @Override  
 public void save(UserExt userExt) {  
 *//取出Ext对象中的所有参数* String username = userExt.getUsername();  
 String password = userExt.getPassword();  
 String email = userExt.getEmail();  
 String phone = userExt.getPhone();  
 String gender = userExt.getGender();  
 Date birthday = userExt.getBirthday();  
 String avatar = userExt.getAvatar();  
  
 *//检查账号是否已经存在  
 //创建一个查询条件模型对象* LambdaQueryWrapper<User> lqw = new LambdaQueryWrapper<>();  
 lqw.eq(User::getUsername, username);  
 *//调用Dao层方法查询* User record = userMapper.selectOne(lqw);  
 *//判断查询结果,若不为空，则用户名已被别人使用，注册失败* if(record != null){  
 throw new CmsException(ResultCode.*USER\_HAS\_EXISTED*);  
 }  
  
 *//对用户提交的密码加密(MD5加密)  
 //password = DigestUtils.md5DigestAsHex(password.getBytes());* password = securityUtil.bcryptEncode(password);  
  
  
 *//把参数封装为一个Entity对象，调用Dao层实现插入  
 //账号未被占用，可以注册  
 //创建一个User对象，把参数封装进去* User user = new User();  
  
 */\* 1）用户在界面上填写、提交的字段信息 \*/* user.setUsername(username);  
 user.setPassword(password);  
 user.setEmail(email);  
 user.setPhone(phone);  
 user.setGender(gender);  
 user.setBirthday(birthday);  
 *//头像* user.setAvatar(avatar);  
  
 */\* 2）需要系统自动生成的字段信息 \*/  
 //设置注册时间为当前时间* user.setRegisterTime(new Date());  
 *//设置状态为启用* user.setStatus("启用");  
 *//设置角色ID默认为3（普通用户）* user.setRoleId(3);  
 *//设置vip状态为0（普通用户）* user.setVip(0);  
 *//设置删除状态默认为0（未删除）* user.setDeleted(0);  
  
 *//把封装好的用户对象，传入Dao层执行数据的插入* userMapper.insert(user);  
 }  
  
 *//根据id查询用户信息* @Override  
 public UserExt getById(Long id) {  
 return UserExt.*toExt*(userMapper.selectById(id));  
 }  
  
 @Override  
 public UserExt getByIdNullable(Long id) {  
 return null;  
 }  
  
 *//修改用户信息* @Override  
 public void update(UserExt userExt) {  
 *//取出请求参数  
 //前端自动提交的参数* Long id = userExt.getId();  
  
 *//用户在页面上填写的信息* String avatar = userExt.getAvatar();  
 String username = userExt.getUsername();  
 String email = userExt.getEmail();  
 String phone = userExt.getPhone();  
 String gender = userExt.getGender();  
 Date birthday = userExt.getBirthday();  
  
 *//创建一个User对象，用于封装用户信息* User user = new User();  
 user.setId(id);  
 user.setAvatar(avatar);  
 user.setUsername(username);  
 user.setEmail(email);  
 user.setPhone(phone);  
 user.setGender(gender);  
 user.setBirthday(birthday);  
  
 *//把封装好的用户对象，传入Dao层执行数据的修改* userMapper.updateById(user);  
  
 }  
  
 *//删除单条数据和多条数据都所有这个方法* @Override  
 public void delete(List<Long> ids) {  
 *//直接调用MybatisPlus提供的方法* userMapper.deleteBatchIds(ids);  
 }  
  
 *//搜索* @Override  
 public IPage<UserExt> pageQueryByClause(UserExt userExt, IPage<User> page) {  
 *//设置查询条件* LambdaQueryWrapper<User> lqw = new LambdaQueryWrapper<>();  
 *//用户名：模糊匹配* lqw.like(StringUtils.*hasText*(userExt.getUsername()),User::getUsername, userExt.getUsername());  
 *//角色* lqw.eq(userExt.getRoleId() != null,User::getRoleId, userExt.getRoleId());  
 *//VIP状态* lqw.eq(userExt.getVip() != null,User::getVip, userExt.getVip());  
 *//账号状态* lqw.eq(StringUtils.*hasText*(userExt.getStatus()),User::getStatus, userExt.getStatus());  
  
 *//执行分页查询操作* userMapper.selectPage(page,lqw);  
  
 *//数据转换* IPage<UserExt> newPage = PageUtil.*convert*(page, UserExt::*toExt*);  
 return newPage;  
 }  
  
 *//* @Override  
 public UserExt getByUsername(String username) {  
 return null;  
 }  
  
 *//查询所有用户信息，用于在资讯模块中使用* @Override  
 public List<UserExt> list() {  
 return UserExt.*toExt*(userMapper.selectList(null));  
 }  
}

CategoryServiceImpl.java

package com.briup.cms.service.impl;  
  
import com.baomidou.mybatisplus.core.conditions.query.LambdaQueryWrapper;  
import com.baomidou.mybatisplus.core.metadata.IPage;  
import com.briup.cms.common.exception.CmsException;  
import com.briup.cms.common.model.entity.Article;  
import com.briup.cms.common.model.entity.Category;  
import com.briup.cms.common.model.entity.User;  
import com.briup.cms.common.model.ext.CategoryExt;  
import com.briup.cms.common.util.PageUtil;  
import com.briup.cms.common.util.ResultCode;  
import com.briup.cms.dao.ArticleMapper;  
import com.briup.cms.dao.CategoryMapper;  
import com.briup.cms.dao.CommentMapper;  
import com.briup.cms.dao.UserMapper;  
import com.briup.cms.service.CategoryService;  
import lombok.RequiredArgsConstructor;  
import org.springframework.stereotype.Service;  
  
import java.io.InputStream;  
import java.io.OutputStream;  
import java.util.List;  
  
*/\*\*  
 \* @author YuYan  
 \* @date 2024-01-09 19:43:02  
 \*/*@Service  
@RequiredArgsConstructor  
public class CategoryServiceImpl implements CategoryService {  
  
 *//用户模块dao层对象* private final UserMapper userMapper;  
  
 *//资讯模块dao层对象* private final ArticleMapper articleMapper;  
  
 *//栏目模块Dao层对象* private final CategoryMapper categoryMapper;  
  
 @Override  
 public void save(CategoryExt categoryExt) {  
 *//取出请求参数* String name = categoryExt.getName();  
 String description = categoryExt.getDescription();  
 Integer orderNum = categoryExt.getOrderNum();  
 Integer parentId = categoryExt.getParentId();  
  
 *//检查栏目名称是否重复  
 //创建条件模型对象* LambdaQueryWrapper<Category> lqw = new LambdaQueryWrapper<>();  
 *//查询条件：栏目名称必须等于给定的name值* lqw.eq(Category::getName, name);  
 *//执行查询* if (categoryMapper.selectOne(lqw) != null) {  
 throw new CmsException(ResultCode.*CATEGORY\_HAS\_EXISTED*);  
 }  
  
 *//参数封装为Entity对象* Category category = new Category();  
 *//用户在界面上填写的信息* category.setName(name);  
 category.setDescription(description);  
 category.setOrderNum(orderNum);  
 category.setParentId(parentId);  
 *//需要系统自动生成的字段* category.setDeleted(0);  
  
 *//调用Dao层执行插入* categoryMapper.insert(category);  
 }  
  
  
 *//根据id删除栏目信息（有特定的要求）  
 //若不允许删除，禁止抛出异常，把能删的都删掉  
 //1、父栏目里有子栏目则不能删除  
 //2、子栏目里存在可用的资讯信息存在（作者信息存在），则禁止删除* @Override  
 public void delete(List<Integer> ids) {  
 *//设置* int deletedCount = 0;  
  
 *//遍历整个集合，对要删除的栏目判断* for (Integer id : ids) {  
 *//根据id，查出栏目的详细信息* Category category = categoryMapper.selectById(id);  
  
 if (category.getParentId() == null) {  
 *//删除父栏目  
 //调用DAO层根据pid查询栏目信息* LambdaQueryWrapper<Category> lqw = new LambdaQueryWrapper<>();  
 lqw.eq(Category::getParentId, id);  
 List<Category> categories = categoryMapper.selectList(lqw);  
 if (categories == null || categories.size() == 0) {  
 *//没有子栏目，可以删除* deletedCount += categoryMapper.deleteById(id);  
 }  
 } else {  
 *//删除子栏目  
 //创建条件模型* LambdaQueryWrapper<Article> lqw = new LambdaQueryWrapper<>();  
 *//设置查询条件：所属分类id* lqw.eq(Article::getCategoryId, id);  
 *//调用到、层执行查询* List<Article> articles = articleMapper.selectList(lqw);  
 *//设置一个变量，表示是否允许删除* boolean deleted = true;  
  
 if (articles != null) {  
 for (Article article : articles) {  
 *//取出作者id* Long userId = article.getUserId();  
 *//查出作者* User user = userMapper.selectById(userId);  
 *//判断是否已注销* if (user.getDeleted().equals(0)) {  
 deleted = false;  
 *//只要找到一个未删除的用户信息，则结束循环* break;  
 }  
 }  
 }  
 *//判断删除标志位，是否允许删除* if (deleted) {  
 deletedCount += categoryMapper.deleteById(id);  
 }  
 }  
 }  
  
 *//抛出异常* if (deletedCount == 0) {  
 throw new CmsException(ResultCode.*CATEGORY\_DELETE\_FAILED*);  
 }  
 }  
  
  
 *//修改栏目信息* @Override  
 public void update(CategoryExt categoryExt) {  
 *//取出参数* Integer id = categoryExt.getId();  
 String name = categoryExt.getName();  
 String description = categoryExt.getDescription();  
 Integer orderNum = categoryExt.getOrderNum();  
 Integer parentId = categoryExt.getParentId();  
  
  
 *//检查栏目名称是否重复  
 //创建条件模型对象* LambdaQueryWrapper<Category> lqw = new LambdaQueryWrapper<>();  
 *//查询条件：栏目名称* lqw.eq(Category::getName, name);  
 *//执行查询，查到的结果结果不为空则报错* Category record = categoryMapper.selectOne(lqw);  
 if (record != null && !record.getId().equals(id)) {  
 throw new CmsException(ResultCode.*CATEGORY\_HAS\_EXISTED*);  
 }  
  
 *//如果当前栏目是父栏目，不允许改成子栏目  
 //调用Dao层根据id查询出这条数据的当前状态* record = categoryMapper.selectById(id);  
 *//如果当前栏目是父栏目，改成子栏目，则报错* if (record.getParentId() == null && parentId != null) {  
 throw new CmsException(ResultCode.*CATEGORY\_LEVEL\_SETTING\_ERROR*);  
 }  
  
 *//把参数封装为entity对象* Category category = new Category();  
 category.setId(id);  
 category.setName(name);  
 category.setDescription(description);  
 category.setOrderNum(orderNum);  
 category.setParentId(parentId);  
  
 *//调用dao层执行修改操作* categoryMapper.updateById(category);  
  
 }  
  
 *//根据id查询栏目信息* @Override  
 public CategoryExt getById(Integer id, boolean cascadeChildren) {  
 *//直接调用方法根据主键查询数据* Category category = categoryMapper.selectById(id);  
  
 *// 将Entity类型转换为Ext类型* CategoryExt categoryExt = CategoryExt.*toExt*(category);  
 return categoryExt;  
 }  
  
 *//根据特定的模式查询栏目信息* @Override  
 public List<CategoryExt> list(String type, boolean cascadeChildren) {  
 *//创建一个查询条件模型对象* LambdaQueryWrapper<Category> lqw = new LambdaQueryWrapper<>();  
 *//判断查询类型type是父栏目还是子栏目* if(type.equals("parent")){  
 *//如果是父栏目，则查询parentId为null的栏目信息* lqw.isNull(Category::getParentId); *//sql where type is null* }else if (type.equals("child")){  
 *//如果是子栏目，则查询parentId不为null的栏目信息* lqw.isNotNull(Category::getParentId); *//sql where type is not null* }  
 *//执行查询* List<Category> categories = categoryMapper.selectList(lqw);  
  
 *//转换数据类型* List<CategoryExt> categoryExts = CategoryExt.*toExt*(categories);  
  
 *//是否查询并封装子级数据* return categoryExts;  
 }  
  
 @Override  
 public IPage<CategoryExt> pageQueryByClause(IPage<Category> page, CategoryExt categoryExt) {  
 *//取出查询条件参数* Integer parentId = categoryExt.getParentId();  
  
 *//设置查询条件  
 //创建一个查询条件模型对象* LambdaQueryWrapper<Category> lqw = new LambdaQueryWrapper<>();  
 *//查询条件：父栏目ID* lqw.eq(parentId != null, Category::getParentId, parentId);  
  
 *//调用dao层执行查询* categoryMapper.selectPage(page, lqw);  
  
 *//转换数据类型* return PageUtil.*convert*(page,CategoryExt::*toExt*);  
  
 }  
  
 @Override  
 public void upload(InputStream is) {  
  
 }  
  
 @Override  
 public void download(OutputStream os) {  
  
 }  
}

ArticleServiceImpl.java

package com.briup.cms.service.impl;  
  
import com.baomidou.mybatisplus.core.conditions.query.LambdaQueryWrapper;  
import com.baomidou.mybatisplus.core.metadata.IPage;  
import com.briup.cms.common.model.entity.Article;  
import com.briup.cms.common.model.ext.ArticleExt;  
import com.briup.cms.common.util.PageUtil;  
import com.briup.cms.common.util.RequestInfoHolder;  
import com.briup.cms.dao.ArticleMapper;  
import com.briup.cms.service.ArticleService;  
import lombok.RequiredArgsConstructor;  
import org.springframework.stereotype.Service;  
import org.springframework.util.StringUtils;  
  
import java.util.Date;  
import java.util.List;  
  
*/\*\*  
 \* 资讯模块  
 \* @author YuYan  
 \* @date 2024-01-09 19:42:29  
 \*/*@Service  
@RequiredArgsConstructor  
public class ArticleServiceImpl implements ArticleService {  
 *// 注入DAO层资讯模块Mapper* private final ArticleMapper articleMapper;  
  
 *//新增或修改资讯信息  
 //若参数中包含参数的id，则修改  
 //若参数中不包含参数的id，则新增* @Override  
 public void saveOrUpdate(ArticleExt articleExt) {  
 *//取出参数* Long id = articleExt.getId();  
 String title = articleExt.getTitle();  
 Integer categoryId = articleExt.getCategoryId();  
 Integer charged = articleExt.getCharged();  
 String content = articleExt.getContent();  
  
 *//获取当前用户id（作者）* Long userId = RequestInfoHolder.*getLong*("userId");  
  
 *//把参数封装为Entity对象* Article article = new Article();  
 article.setId(id);  
 */\* 用户在页面上填写的字段 \*/* article.setTitle(title);  
 article.setCategoryId(categoryId);  
 article.setCharged(charged);  
 article.setContent(content);  
  
  
 *//判断是新增还是修改* if (id != null) {  
 */\* 修改操作 \*/  
 //如果当前修改操作是修改未审核，则修改完为未审核状态  
 //如果当前审核不通过，修改完则为未审核状态* article.setStatus("未审核");  
 *// 调用Dao层执行插入即可* articleMapper.updateById(article);  
 return;  
 }  
 */\* 新增操作 \*/* article.setUserId(userId);  
 */\* 新增文章的时候系统自动生成的字段 \*/  
 // 资讯刚发表时都默认设为未审核状态* article.setStatus("未审核");  
 *// 系统自动设置资讯的发布时间为当前时间* article.setPublishTime(new Date());  
 *// 逻辑删除标志位* article.setDeleted(0);  
 *// 阅读量、点赞量、点踩量都设为0* article.setReadNum(0);  
 article.setLikeNum(0);  
 article.setDislikeNum(0);  
 *// 调用Dao层执行插入* articleMapper.insert(article);  
 }  
  
  
 *//审核资讯* @Override  
 public void review(Long id, String status) {  
 *//把参数id、status封装为Entity对象* Article article = new Article();  
 article.setId(id);  
 article.setStatus(status);  
 *//调用Dao层执行更新操作* articleMapper.updateById(article);  
 }  
  
 *//删除资讯* @Override  
 public void delete(List<Long> ids) {  
 articleMapper.deleteBatchIds(ids);  
 }  
  
 *//根据id查询资讯信息* @Override  
 public ArticleExt getById(Long id, int commentsNum) {  
 ArticleExt articleExt = ArticleExt.*toExt*(articleMapper.selectById(id));  
 return articleExt;  
 }  
  
 *//分页+多条件查询资讯信息* @Override  
 public IPage<ArticleExt> pageQueryByClause(IPage<Article> page, ArticleExt articleExt) {  
 *//取出参数* String title = articleExt.getTitle();  
 String status = articleExt.getStatus();  
 Long userId = articleExt.getUserId();  
 Integer charged = articleExt.getCharged();  
 Date startTime = articleExt.getStartTime();  
 Date endTime = articleExt.getEndTime();  
 Integer categoryId = articleExt.getCategoryId();  
  
 *//参数封装为条件模型对象* LambdaQueryWrapper<Article> lqw = new LambdaQueryWrapper<>();  
 *//文章标题，模糊匹配* lqw.like(StringUtils.*hasText*(title), Article::getTitle, title);  
 *//所属栏目，精准匹配* lqw.eq(categoryId != null, Article::getCategoryId, categoryId);  
 *//审核状态，精准匹配* lqw.eq(StringUtils.*hasText*(status), Article::getStatus, status);  
 *//发布人，精准匹配* lqw.eq(userId != null, Article::getUserId, userId);  
 *//是否收费，精准匹配* lqw.eq(charged != null, Article::getCharged, charged);  
 *//开始时间，范围匹配（发布时间晚于限定的开始时间）  
 //gt = greater than 大于* lqw.ge(startTime != null, Article::getPublishTime, startTime);  
 *//结束时间，范围匹配  
 //lt = less than 小于* lqw.le(endTime != null, Article::getPublishTime, endTime);  
  
 *//执行分页查询* articleMapper.selectPage(page, lqw);  
  
 *//转换数据类型* IPage<ArticleExt> newPage = PageUtil.*convert*(page, ArticleExt::*toExt*);  
  
 *//...* return newPage;  
 }  
  
 *//查询所有资讯信息（用于评论模块）* @Override  
 public List<ArticleExt> list() {  
 return ArticleExt.*toExt*(articleMapper.selectList(null));  
 }  
}

CommentServiceImpl.java

package com.briup.cms.service.impl;  
  
import com.baomidou.mybatisplus.core.conditions.query.LambdaQueryWrapper;  
import com.baomidou.mybatisplus.core.metadata.IPage;  
import com.briup.cms.common.model.entity.Comment;  
import com.briup.cms.common.model.entity.SubComment;  
import com.briup.cms.common.model.ext.CommentExt;  
import com.briup.cms.common.model.ext.SubCommentExt;  
import com.briup.cms.common.model.ext.UserExt;  
import com.briup.cms.common.model.param.CommentDeleteParam;  
import com.briup.cms.common.util.PageUtil;  
import com.briup.cms.dao.CommentMapper;  
import com.briup.cms.dao.SubCommentMapper;  
import com.briup.cms.dao.UserMapper;  
import com.briup.cms.service.CommentService;  
import lombok.RequiredArgsConstructor;  
import org.springframework.stereotype.Service;  
import org.springframework.util.StringUtils;  
  
import java.util.Date;  
import java.util.List;  
  
*/\*\*  
 \* 评论模块(Comment)表服务实现类  
 \* @author YuYan  
 \* @date 2024-01-09 19:42:59  
 \*/*@Service  
@RequiredArgsConstructor  
public class CommentServiceImpl implements CommentService {  
 *// 一级评论* private final CommentMapper commentMapper;  
 *//二级评论* private final SubCommentMapper subCommentMapper;  
 *//用户模块* private final UserMapper userMapper;  
  
  
 @Override  
 public void save(CommentExt commentExt) {  
  
 }  
  
 @Override  
 public void save(SubCommentExt commentExt) {  
  
 }  
  
  
 *//删除评论* @Override  
 public void delete(Long id, String type) {  
 if (type.equals("parent")) {  
 *//删除一级评论  
 //创建一个条件模型对象* LambdaQueryWrapper<SubComment> lqw = new LambdaQueryWrapper<>();  
 *//设置删除条件parent\_id必须等于当前要删除的一级评论的id* lqw.eq(SubComment::getParentId, id);  
 *//执行删除操作（二级评论）* subCommentMapper.delete(lqw);  
 *//删除一级评论* commentMapper.deleteById(id);  
 } else if (type.equals("child")) {  
 *//删除二级评论* subCommentMapper.deleteById(id);  
 }  
 }  
  
 @Override  
 public void deleteBatch(List<CommentDeleteParam> params) {  
  
 }  
  
 @Override  
 public List<SubCommentExt> list(SubCommentExt subCommentParam) {  
 return null;  
 }  
  
 *//分类+多条件检索评论信息* @Override  
 public IPage<CommentExt> pageQueryByClause(IPage<Comment> page, CommentExt commentParam) {  
 *//取出参数* Long userId = commentParam.getUserId();  
 Long articleId = commentParam.getArticleId();  
 String content = commentParam.getContent();  
 Date startTime = commentParam.getStartTime();  
 Date endTime = commentParam.getEndTime();  
  
 *//设置查询条件* LambdaQueryWrapper<Comment> lqw = new LambdaQueryWrapper<>();  
 lqw.eq(userId != null, Comment::getUserId, userId);  
 lqw.eq(articleId != null, Comment::getArticleId, articleId);  
 lqw.like(StringUtils.*hasText*(content), Comment::getContent, content);  
 *//ge greater than or equal大于或等于  
 //le less than or equal小于或等于* lqw.ge(startTime != null, Comment::getPublishTime, startTime);  
 lqw.le(endTime != null, Comment::getPublishTime, endTime);  
  
 *//调用dao层执行查询* commentMapper.selectPage(page, lqw);  
  
 *//数据类型转换* IPage<CommentExt> newPage = PageUtil.*convert*(page, CommentExt::*toExt*);  
  
 *//关联查询每条评论的作者信息* List<CommentExt> records = newPage.getRecords();  
 *//遍历这个集合* for (CommentExt record : records) {  
 *//从一级评论取出评论作者id* Long authorId = record.getUserId();  
 *//查询用户详细信息* UserExt userExt = UserExt.*toExt*(userMapper.selectById(authorId));  
 *//* record.setUserExt(userExt);  
  
 *//创建查询二级评论的条件模型对象* LambdaQueryWrapper<SubComment> subLqw = new LambdaQueryWrapper<>();  
 *//设置查询条件:parent\_id等于一级评论对象id、* subLqw.eq(SubComment::getParentId, record.getId());  
 *//执行查询* List<SubComment> subComments = subCommentMapper.selectList(subLqw);  
 *//转换数据类型* List<SubCommentExt> subCommentExts = SubCommentExt.*toExt*(subComments);  
 *//遍历每个二级评论，查询发布者信息* for (SubCommentExt subCommentExt : subCommentExts) {  
 *//从二级评论取出评论作者id* Long subAuthorId = record.getUserId();  
 *//查询用户详细信息* userExt = UserExt.*toExt*(userMapper.selectById(subAuthorId));  
 *//* subCommentExt.setUserExt(userExt);  
 }  
 *//设置到一级评论对象中* record.setSubCommentExts(subCommentExts);  
  
 }  
  
 return newPage;  
 }  
}

RoleServiceImpl.java

package com.briup.cms.service.impl;  
  
import com.briup.cms.common.model.ext.RoleExt;  
import com.briup.cms.dao.RoleMapper;  
import com.briup.cms.service.RoleService;  
import lombok.RequiredArgsConstructor;  
import org.springframework.stereotype.Service;  
  
import java.util.List;  
  
*/\*\*  
 \* 业务逻辑层实现类 - 角色相关  
 \* @author YuYan  
 \* @date 2024-01-09 19:42:53  
 \*/*@Service  
@RequiredArgsConstructor  
public class RoleServiceImpl implements RoleService {  
 *//注入持久层对象* private final RoleMapper roleMapper;  
  
 *//根据id查询角色信息* @Override  
 public RoleExt getById(Integer id) {  
 *//查询单条数据selectById()* return RoleExt.*toExt*(roleMapper.selectById(id));  
 }  
  
 *//查询所有角色信息* @Override  
 public List<RoleExt> list() {  
 *//调用selectList()方法，用于批量查询数据传入null表示查询全部数据* return RoleExt.*toExt*(roleMapper.selectList(null));

package com.briup.cms.web.controller;  
  
import com.baomidou.mybatisplus.core.metadata.IPage;  
import com.baomidou.mybatisplus.extension.plugins.pagination.Page;  
import com.briup.cms.common.log.LogAccess;  
import com.briup.cms.common.model.entity.Article;  
import com.briup.cms.common.model.ext.ArticleExt;  
import com.briup.cms.common.model.vo.ArticleVO;  
import com.briup.cms.common.util.PageUtil;  
import com.briup.cms.common.util.Result;  
import com.briup.cms.service.ArticleService;  
import lombok.RequiredArgsConstructor;  
import org.springframework.web.bind.annotation.\*;  
  
import java.util.Date;  
import java.util.List;  
  
*/\*\*  
 \* 资讯相关功能  
 \* @author YuYan  
 \* @date 2023-12-05 11:29:43  
 \*/*@RestController  
@RequiredArgsConstructor  
@RequestMapping("/auth/article")  
public class ArticleController {  
  
 private final ArticleService articleService;  
  
 */\*\*  
 \* 查询所有资讯信息  
 \* @return  
 \*/* @GetMapping  
 public Result list() {  
 return Result.*ok*(ArticleVO.*toVO*(articleService.list()));  
 }  
  
 */\*\*  
 \* 分页+条件检索资讯信息  
 \* @param pageNum 当前页  
 \* @param pageSize 每页大小  
 \* @param title 标题关键字  
 \* @param categoryId 所属栏目ID  
 \* @param status 审核状态  
 \* @param userId 发布者ID  
 \* @param charged 收费状态  
 \* @param startTime 开始时间  
 \* @param endTime 结束时间  
 \* @return  
 \*/* @GetMapping(params = "page=true")  
 public Result pageQuery(@RequestParam(value = "pageNum", required = true) int pageNum,  
 @RequestParam(value = "pageSize", required = true) int pageSize,  
 @RequestParam(value = "title", required = false) String title,  
 @RequestParam(value = "categoryId", required = false) Integer categoryId,  
 @RequestParam(value = "status", required = false) String status,  
 @RequestParam(value = "userId", required = false) Long userId,  
 @RequestParam(value = "charged", required = false) Integer charged,  
 @RequestParam(value = "startTime", required = false) Date startTime,  
 @RequestParam(value = "endTime", required = false) Date endTime) {  
 */\* 封装分页参数 \*/* IPage<Article> page = new Page<>(pageNum, pageSize);  
 */\* 封装查询条件 \*/* ArticleExt articleExt = ArticleExt.*builder*()  
 .title(title)  
 .categoryId(categoryId)  
 .status(status)  
 .userId(userId)  
 .charged(charged)  
 .startTime(startTime)  
 .endTime(endTime)  
 .build();  
 return Result.*ok*(PageUtil.*convert*(  
 articleService.pageQueryByClause(page, articleExt),  
 ArticleVO::*toVO*));  
 }  
  
 */\*\*  
 \* 新增或修改资讯信息  
 \* @param articleVO  
 \* @return  
 \*/* @PostMapping  
 @LogAccess("新增或修改资讯信息")  
 public Result save(@RequestBody ArticleVO articleVO) {  
 articleService.saveOrUpdate(ArticleExt.*toExt*(articleVO));  
 return Result.*ok*();  
 }  
  
 */\*\*  
 \* 审核资讯  
 \* @param id 文章ID  
 \* @param status 审核状态值（审核通过、审核不通过）  
 \* @return  
 \*/* @PutMapping("/{id}")  
 @LogAccess("审核资讯")  
 public Result review(@PathVariable(value = "id") Long id,  
 @RequestBody(required = false) String status) {  
 articleService.review(id, status);  
 return Result.*ok*();  
 }  
  
 */\*\*  
 \* 删除资讯信息  
 \* @param ids  
 \* @return  
 \*/* @DeleteMapping("/{ids}")  
 @LogAccess("删除资讯信息")  
 public Result delete(@PathVariable(value = "ids") List<Long> ids) {  
 articleService.delete(ids);  
 return Result.*ok*();  
 }  
  
 */\*\*  
 \* 根据ID查询资讯信息  
 \* @param id  
 \* @return  
 \*/* @GetMapping("/{id}")  
 public Result getById(@PathVariable(value = "id") Long id) {  
 return Result.*ok*(ArticleVO.*toVO*(articleService.getById(id, 3)));  
 }  
  
}

package com.briup.cms.web.controller;  
  
import com.briup.cms.common.model.vo.UserVO;  
import com.briup.cms.common.util.GlobalConstants;  
import com.briup.cms.common.util.Result;  
import com.briup.cms.service.AuthService;  
import lombok.RequiredArgsConstructor;  
import org.springframework.web.bind.annotation.\*;  
  
*/\*\*  
 \* 认证相关功能  
 \* @author YuYan  
 \* @date 2023-12-01 14:21:51  
 \*/*@RestController  
@RequiredArgsConstructor  
public class AuthController {  
  
 private final AuthService authService;  
  
 */\*\*  
 \* 用户登录  
 \* @param userVO  
 \* @return  
 \*/* @PostMapping("/login")  
 public Result login(@RequestBody UserVO userVO) {  
 return Result.*ok*(authService.login(  
 userVO.getUsername(),  
 userVO.getPassword()));  
 }  
  
 */\*\*  
 \* 根据Token获取用户信息  
 \* @param token  
 \* @return  
 \*/* @GetMapping("/auth/userinfo")  
 public Result userinfo(@RequestHeader(GlobalConstants.*TOKEN\_HEADER\_NAME*)  
 String token) {  
 return Result.*ok*(UserVO.*toVO*(authService.getUserinfo(token)));  
 }  
  
 */\*\*  
 \* 用户登出  
 \* @return  
 \*/* @PostMapping("/logout")  
 public Result logout() {  
 return Result.*ok*();  
 }  
  
}

package com.briup.cms.web.controller;  
  
import com.baomidou.mybatisplus.core.metadata.IPage;  
import com.baomidou.mybatisplus.extension.plugins.pagination.Page;  
import com.briup.cms.common.download.Download;  
import com.briup.cms.common.exception.CmsException;  
import com.briup.cms.common.log.LogAccess;  
import com.briup.cms.common.model.entity.Category;  
import com.briup.cms.common.model.ext.CategoryExt;  
import com.briup.cms.common.model.vo.CategoryVO;  
import com.briup.cms.common.util.\*;  
import com.briup.cms.service.CategoryService;  
import lombok.RequiredArgsConstructor;  
import org.springframework.web.bind.annotation.\*;  
import org.springframework.web.multipart.MultipartFile;  
  
import javax.servlet.http.HttpServletResponse;  
import java.io.InputStream;  
import java.io.OutputStream;  
import java.util.List;  
  
*/\*\*  
 \* 栏目相关功能  
 \* @author YuYan  
 \* @date 2023-12-04 11:08:51  
 \*/*@RestController  
@RequiredArgsConstructor  
@RequestMapping("/auth/category")  
public class CategoryController {  
  
 private final CategoryService categoryService;  
  
 */\*\*  
 \* 新增栏目  
 \* @param categoryVO  
 \* @return  
 \*/* @LogAccess("新增栏目")  
 @PostMapping  
 public Result save(@RequestBody CategoryVO categoryVO) {  
 */\* 转换参数类型 \*/  
 /\* 调用Service层实现插入 \*/* categoryService.save(CategoryExt.*toExt*(categoryVO));  
 return Result.*ok*();  
 }  
  
 */\*\*  
 \* 删除栏目信息  
 \* @param ids  
 \* @return  
 \*/* @DeleteMapping("/{ids}")  
 public Result delete(@PathVariable("ids") List<Integer> ids) {  
 categoryService.delete(ids);  
 return Result.*ok*();  
 }  
  
 */\*\*  
 \* 修改栏目信息  
 \* @param categoryVO  
 \* @return  
 \*/* @PutMapping  
 public Result update(@RequestBody CategoryVO categoryVO) {  
 */\* 转换参数类型 \*/  
 /\* 调用Service实现修改 \*/* categoryService.update(CategoryExt.*toExt*(categoryVO));  
 return Result.*ok*();  
 }  
  
 */\*\*  
 \* 分页+条件检索栏目信息  
 \* @param pageNum 当前页  
 \* @param pageSize 每页大小  
 \* @param parentId 父栏目ID  
 \* @return  
 \*/* @GetMapping(params = "page=true")  
 public Result pageQuery(@RequestParam(value = "pageNum", required = true) int pageNum,  
 @RequestParam(value = "pageSize", required = true) int pageSize,  
 @RequestParam(value = "parentId", required = false) Integer parentId) {  
 */\* 封装参数 \*/* IPage<Category> page = new Page<>(pageNum, pageSize);  
 CategoryExt categoryExt = CategoryExt.*builder*()  
 .parentId(parentId)  
 .build();  
 */\* 调用Service实现查询 \*/* return Result.*ok*(PageUtil.*convert*(  
 categoryService.pageQueryByClause(page, categoryExt),  
 CategoryVO::*toVO*));  
 }  
  
 */\*\*  
 \* 根据ID查询栏目信息  
 \* @param id 栏目ID  
 \* @return  
 \*/* @GetMapping("/{id}")  
 public Result getById(@PathVariable(value = "id") Integer id,  
 @RequestParam(value = "cascadeChildren", required = false, defaultValue = "false") boolean cascadeChildren) {  
 return Result.*ok*(CategoryVO.*toVO*(categoryService.getById(id, cascadeChildren)));  
 }  
  
 */\*\*  
 \* 查询栏目信息  
 \* @param cascadeChildren 是否要关联查询二级栏目  
 \* @return  
 \*/* @GetMapping  
 public Result list(@RequestParam(value = "type", required = false) String type,  
 @RequestParam(value = "cascadeChildren", required = false, defaultValue = "false") boolean cascadeChildren) {  
 return Result.*ok*(CategoryVO.*toVO*(  
 categoryService.list(type, cascadeChildren)));  
 }  
  
 */\*\*  
 \* 用于下载导入栏目使用的Excel模板文件  
 \* 不再像其他接口一样返回Result格式的响应对象，  
 \* 而是要手动操作Response对象写回数据，所以返回值设为void即可。  
 \*/* @GetMapping(params = {"action=download", "type=template"})  
 @Download(fileName = "栏目模板.xlsx")  
 public void downloadTemplate(HttpServletResponse response) throws Exception {  
 */\* 获取相应输出流对象 \*/* OutputStream os = response.getOutputStream();  
 */\* 获取读取文件的输入流对象 \*/* InputStream is = ClassLoader.*getSystemResourceAsStream*("category\_template.xlsx");  
 if (ObjectUtil.*isNull*(is)) {  
 throw new CmsException(ResultCode.*CATEGORY\_TEMPLATE\_FILE\_NOT\_EXIST*);  
 }  
 IOUtil.*copyData*(is, os);  
 }  
  
 */\*\*  
 \* 导出栏目数据为Excel  
 \* @return  
 \* @throws Exception  
 \*/* @GetMapping(params = {"action=download", "type=data"})  
 @Download(fileName = "栏目列表.xlsx")  
 public void downloadData(HttpServletResponse response) throws Exception {  
 categoryService.download(response.getOutputStream());  
 }  
  
 */\*\*  
 \* 通过Excel上传栏目信息  
 \* @param multipartFile  
 \* @return  
 \* @throws Exception  
 \*/* @PostMapping(params = "action=upload")  
 public Result upload(@RequestParam("file") MultipartFile multipartFile) throws Exception {  
 categoryService.upload(multipartFile.getInputStream());  
 return Result.*ok*();  
 }  
  
  
}

package com.briup.cms.web.controller;  
  
import com.baomidou.mybatisplus.core.metadata.IPage;  
import com.baomidou.mybatisplus.extension.plugins.pagination.Page;  
import com.briup.cms.common.download.Download;  
import com.briup.cms.common.model.entity.Log;  
import com.briup.cms.common.model.ext.LogExt;  
import com.briup.cms.common.model.vo.LogEntityVO;  
import com.briup.cms.common.util.PageUtil;  
import com.briup.cms.common.util.Result;  
import com.briup.cms.service.LogService;  
import lombok.RequiredArgsConstructor;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.RequestMapping;  
import org.springframework.web.bind.annotation.RequestParam;  
import org.springframework.web.bind.annotation.RestController;  
  
import javax.servlet.http.HttpServletResponse;  
import java.util.Date;  
  
*/\*\*  
 \* @author YuYan  
 \* @date 2023-12-11 09:52:06  
 \*/*@RestController  
@RequiredArgsConstructor  
@RequestMapping("/auth/log")  
public class LogController {  
  
 */\*\*  
 \* 日志模块Service层对象  
 \*/* private final LogService logService;  
  
 @GetMapping(params = "page=true")  
 public Result pageQuery(@RequestParam(value = "pageSize", required = true) int pageSize,  
 @RequestParam(value = "pageNum", required = true) int pageNum,  
 @RequestParam(value = "username", required = false) String username,  
 @RequestParam(value = "requestUrl", required = false) String requestUrl,  
 @RequestParam(value = "startTime", required = false) Date startTime,  
 @RequestParam(value = "endTime", required = false) Date endTime) {  
  
 IPage<Log> page = new Page<>(pageNum, pageSize);  
  
 LogExt logExt = LogExt.*builder*()  
 .username(username)  
 .requestUrl(requestUrl)  
 .startTime(startTime)  
 .endTime(endTime)  
 .build();  
  
 return Result.*ok*(PageUtil.*convert*(logService.pageQueryByClause(page, logExt), LogEntityVO::*toVO*));  
 }  
  
 @GetMapping(params = {"action=download", "type=data"})  
 @Download(fileName = "日志.xlsx")  
 public void downloadData(HttpServletResponse response,  
 @RequestParam(value = "username", required = false) String username,  
 @RequestParam(value = "requestUrl", required = false) String requestUrl,  
 @RequestParam(value = "startTime", required = false) Date startTime,  
 @RequestParam(value = "endTime", required = false) Date endTime,  
 @RequestParam(value = "count", required = false, defaultValue = "-1") int limit) throws Exception {  
 */\* 封装请求参数 \*/* logService.download(response.getOutputStream(),  
 LogExt.*builder*()  
 .username(username)  
 .requestUrl(requestUrl)  
 .startTime(startTime)  
 .endTime(endTime)  
 .limit(limit)  
 .build());  
 }  
  
}

package com.briup.cms.web.controller;  
  
import com.briup.cms.common.model.vo.RoleVO;  
import com.briup.cms.common.util.Result;  
import com.briup.cms.service.RoleService;  
import lombok.RequiredArgsConstructor;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.RequestMapping;  
import org.springframework.web.bind.annotation.RestController;  
  
*/\*\*  
 \* 角色相关功能  
 \* @author YuYan  
 \* @date 2023-12-01 17:01:31  
 \*/*@RestController  
@RequiredArgsConstructor  
@RequestMapping("/role")  
public class RoleController {  
  
 private final RoleService roleService;  
  
 */\*\*  
 \* 查询所有角色信息  
 \* @return  
 \*/* @GetMapping  
 public Result list() {  
 return Result.*ok*(RoleVO.*toVO*(roleService.list()));  
 }  
  
}