

上海电力大学

Java EE 大作业



题 目： 基于 Java EE 平台的计算机学院

门户网站设计与实现

学 号： 20221560

姓 名： 杨裕凯

院 系： 计算机科学与技术学院

专业年级： 计算机科学与技术专业（计卓班）

2024 年 12 月 27 日

目录

1 绪论	1
1.1 研究目的与意义	1
1.2 主要研究内容	1
2 关键技术介绍	3
2.1 前端技术 JSP	3
2.2 后端处理 Servlet	3
2.3 数据库连接 JDBC	3
2.4 本章小结	4
3 系统需求分析	5
3.1 需求分析概述	5
3.2 功能需求分析	5
3.3 非功能需求分析	6
3.4 本章小结	7
4 系统设计	8
4.1 系统总体设计	8
4.2 系统功能设计	9
4.3 系统数据库设计	9
4.4 本章小结	12
5 系统实现与测试	13
5.1 系统环境	13
5.2 关键功能实现	13
5.3 系统展示	41
5.4 本章小结	47
6 总结与展望	49
6.1 全文总结	49
6.2 未来展望	49

1 绪论

1.1 研究目的与意义

通过开发基于 Java EE 平台的计算机学院门户网站系统，旨在将课堂所学的理论知识转化为实际应用能力。本项目综合运用了 Java Web 开发中的核心技术，包括前端技术（HTML5、CSS3、JavaScript、jQuery）和后端技术（JSP、Servlet、JavaBean、JDBC）等。

在开发过程中，重点掌握了以下技术要点：

- (1) 使用 JSP 实现前端页面的创造
- (2) 运用 Bootstrap 框架实现响应式布局设计
- (3) 使用 Servlet 实现后端功能
- (4) 通过 Ajax 技术实现前后端数据交互
- (5) 应用数据库连接和事务管理确保系统性能和数据一致性

通过本项目的实践，不仅提升了 Java Web 全栈开发能力，也加深了对软件工程开发流程的认识，包括需求分析、系统设计、编码实现、测试部署等各个环节。同时，在项目开发过程中遇到的各种技术难题的解决过程，也培养了独立思考和问题解决能力。这些经验对未来从事相关领域的开发工作具有重要的实践意义。

1.2 主要研究内容

论文拟设计并实现一套面向计算机学院的综合门户网站系统，是学院信息化建设的重要组成部分。该系统的主要功能是为学院提供一个统一的信息发布、资源管理和交互平台，实现学院新闻、教学资源、学术科研等信息的高效管理和展示。系统通过模块化设计，支持新闻资讯发布、教师风采展示、学术成果管理、招生就业信息等多种功能，为师生提供全方位的信息服务。由于系统涉及教师、学生等多类用户的个人信息及学院内部数据，需要建立完善的权限管理机制，确保数据访问的安全性，防止未经授权的访问和信息泄露。

根据业务需求，课题通过将学院实际需求和 Web 开发技术相结合，采用基于 Java EE 的 B/S 架构，对计算机学院门户网站系统进行研究和开发，加深对教育信息化建设的理解，掌握现代 Web 应用开发的完整流程。主要研究内容如下：

- (1) 深入分析学院各部门业务需求，明确系统功能定位，设计合理的信息架构和用户角色权限体系，确定核心功能模块。

- (2) 完成系统总体架构设计，采用前后端分离模式，确定技术栈选型，设计数据库结构，规划系统部署方案，并对各功能模块进行详细设计。
- (3) 基于设计方案，运用 JSP、Servlet 等主流框架技术实现系统功能，包括用户认证、信息管理、资源下载等模块，并通过单元测试、集成测试等手段验证系统功能的完整性。

2 关键技术介绍

本章简要介绍系统开发中运用的相关技术，为开发提供技术支持。主要包括 JSP 页面技术、Servlet 后端处理技术以及 JDBC 数据库连接技术。这些技术构成了 Java Web 开发的核心技术栈，为系统的实现提供了重要保障。

2.1 前端技术 JSP

JSP (JavaServer Pages) 是一种基于 Java 语言的动态网页技术标准。JSP 将 Java 代码和 HTML 标记混合在同一个文件中，以".JSP"为扩展名。JSP 具有以下主要特点：

- (1) JSP 页面本质上是一个 Servlet，在第一次被访问时会被转换为 Servlet 的 Java 源代码。
- (2) 支持 Java 脚本元素，包括声明、表达式和脚本程序片段。
- (3) 提供了九大内置对象 (request、response、session 等)，简化了开发过程。
- (4) 支持自定义标签库 (JSTL)，提高了代码的可重用性和可维护性。

在本系统中，JSP 主要用于实现动态页面的展示，包括新闻列表、通知公告、教务通知等功能模块的前端页面实现。

2.2 后端处理 Servlet

Servlet 是 Java Web 应用的核心组件，是运行在 Web 服务器上的 Java 类，用于处理客户端的请求并动态生成响应内容。Servlet 的主要特点包括：

- (1) 基于 Java 语言开发，继承 HttpServlet 类，具有跨平台特性。
- (2) 采用单实例多线程的工作模式，效率高。
- (3) 生命周期由 Web 容器管理，包括初始化、服务和销毁三个阶段。
- (4) 提供了处理 HTTP 请求的标准接口，如 doGet()和 doPost()方法。

在本系统中，Servlet 主要负责处理用户请求、业务逻辑实现和数据处理等后端功能。

2.3 数据库连接 JDBC

JDBC (Java Database Connectivity) 是 Java 语言中用于执行 SQL 语句的标准 API，它提供了一组用于访问关系数据库的接口。JDBC 的主要特点如下：

- (1) 提供了统一的数据库访问接口，支持多种数据库。
- (2) 实现了数据库连接池技术，提高了系统性能。

(3) 支持事务处理，确保数据的一致性。

(4) 提供了预编译 SQL 语句功能，防止 SQL 注入攻击。

在本系统中，JDBC 主要用于实现与 SQL Server 数据库的连接和数据操作，包括用户信息管理、新闻内容存储、资源信息管理等功能的数据持久化。

2.4 本章小结

本章介绍了系统开发所涉及的三项核心技术：JSP、Servlet 和 JDBC。这些技术相互配合，构成了一个完整的 Web 应用开发体系。JSP 负责页面展示，Servlet 处理业务逻辑，JDBC 实现数据持久化，三者的结合使得系统具备了完整的前端展示、后端处理和数据存储能力。通过对这些技术的合理运用，可以有效地实现计算机学院门户网站的各项功能需求。

3 系统需求分析

在软件工程开发过程中，需求分析是一个至关重要的环节，它直接影响到后续系统设计、开发和测试等各个阶段的质量。准确的需求分析不仅能够指导开发团队精准把握用户需求，也是系统最终验收的重要依据。计算机学院门户网站系统作为学院信息化建设的重要组成部分，承担着信息发布、资源共享、教学管理等多项功能，需要在充分理解学院各部门业务需求的基础上，建立一个功能完善、操作便捷、安全可靠的综合信息服务平台。

3.1 需求分析概述

本章将从系统的功能需求、非功能需求两个维度展开分析。在功能需求方面，重点关注系统的用户角色划分、信息管理流程、资源共享机制等核心功能；在非功能需求方面，着重分析系统的性能要求、安全性需求、可用性需求等技术指标，为后续的系统设计和开发工作奠定基础。通过全面且深入的需求分析，确保系统能够满足学院日常教学、科研、管理等各方面的实际需要。

3.2 功能需求分析

软件面向的群体主要有三种：普通教职工和学生、管理员、游客。他们对系统的功能需求略有不同。游客所需要的是获取学院的一些基本信息，所以他们可以查看大多数展示页面，比如新闻动态、学院简介等。普通教职工和学生则可以查看绝大多数页面，除去游客可以查看的界面之外，还可以查看教务通知、通知公告两个页面，以及可以使用公共服务栏下的所有功能。而对于管理员，则可以实现对新闻的管理、对通知公告的管理以及对资源下载页面的管理。

针对后端数据管理系统，将其分成多个模块，主要包括登录模块、新闻动态模块、公共服务模块，其具体又分成了几个模块，可以参照下图。

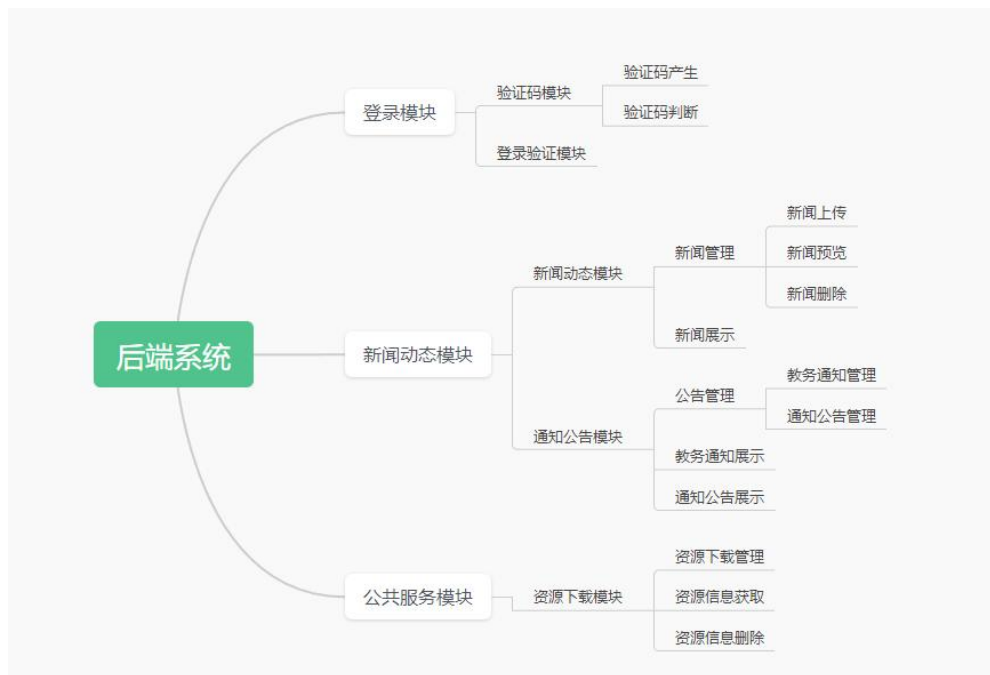


图 1 后端系统模块划分

3.3 非功能需求分析

非功能需求是对系统在功能实现过程中需要遵循的约束条件和质量要求，是确保系统稳定运行的重要保障。本系统主要面向学院师生用户，下面从可靠性、安全性、易用性、可扩展性等几个方面进行分析。

(1) 可靠性

系统部署到服务器上后，需要长时间运行提供服务。运行期间，系统不能经常崩溃、数据丢失或者不一致。故障出现时，要降低错误的危害程度，业务、数据、用户等要隔离，防止错误蔓延，软硬件也要具备故障在线恢复能力。要有一定的容错设计，通过冗余提高资源的可靠性。

(2) 安全性

学院官网信息涉及到学院形象展示，是非常重要的一个公开信息。如果随便被人破解，会对学院形象和信誉造成严重破坏。同时，针对一些文件资源，例如资源下载功能，如果允许任何人进行资源上传，极易造成病毒等的入侵，从而破坏学院服务器，造成极大的损失。因此，在安全性这一块，访问系统需要确认身份；对重要数据，需要加密存储；对系统的不同角色，要进行访问控制。

(3) 易用性

系统的用户一般为非计算机专业人员，并不了解系统的设计实现原理，他们

多限于使用键盘、鼠标、显示器等外设对系统进行操作，因此，界面的简介美观性，操作的简便性，对用户来说是最直接的感受。易用性要求界面简介，功能结构合理，操作流程符合用户习惯，没有乱码等让用户摸不着头脑的错误，导航清晰，版面规则。重要操作给予提醒，错误操作给出明确信息，操作结果要正确响应和反映。

(4) 可拓展性

系统往往不是一成不变的，随着用户的使用和新需求的提出，多需要进行二次、三次开发，对功能进行完善、新增、删除等。因此在设计和开发时，就要考虑到将来的可扩展性，合理划分模块，降低模块、代码之间的耦合度，减少相互之间的依赖，为未来系统的扩展留下一定空间，同时对系统结构和已有代码不会产生太大的影响。

3.4 本章小结

本章对计算机学院门户网站系统进行了全面的需求分析。首先从整体角度阐述了需求分析在系统开发中的重要性，明确了系统作为学院信息化建设重要组成部分的定位。

在功能需求方面，根据用户角色将系统划分为面向游客、普通教职工和学生、管理员三类用户群体，并详细分析了各类用户的功能需求。同时，对系统的后端管理模块进行了清晰的划分，包括登录模块、新闻动态模块、公共服务模块等，为系统的具体实现提供了明确的功能指导。

在非功能需求方面，从可靠性、安全性、易用性和可扩展性四个维度进行了深入分析。重点强调了系统运行的稳定性要求、数据安全保护机制、用户友好的界面设计以及系统未来扩展的可能性，这些要求将作为系统开发过程中的重要约束条件。

通过本章的需求分析，为后续的系统设计和开发工作建立了清晰的目标和框架，确保了系统开发方向与学院实际需求的一致性，为打造一个功能完善、安全可靠、易于使用的学院门户网站奠定了基础。

4 系统设计

本章主要针对整个题目的总体要求进行分析，并在最终同时给出数据库的具体设计。

4.1 系统总体设计

总体设计从宏观层面描述系统的整体布局，这里主要说明逻辑架构。逻辑架构则对软件层次进行划分，明确技术选型，对软件开发有着直接的指导意义。

后台管理系统整体上采用 B/S 体系架构，工作模式为浏览器发送请求，服务器返回响应。在传统的 Java Web 开发模式中，普遍采用 JSP 和 Servlet 的结构，前端 JSP 页面和后端 Servlet 业务逻辑打包在一起，运行在应用服务器的 Tomcat 中。

根据逻辑架构的分层设计图，系统采用经典的三层架构设计，从上到下依次为表示层、业务逻辑层和数据访问层。

表示层是系统的最上层，直接面向最终用户，主要负责处理用户界面的显示和用户输入。本系统的表示层采用 JSP 技术实现，主要包括项目绝大多数的静态页面，例如学院概括、学科建设、本科教学等，以及一些需要依赖 java 脚本实现的一系列前端展示，比如新闻动态等需要从数据库当中获取的动态页面。

业务逻辑层是系统的核心层，负责实现具体的业务逻辑处理。该层通过 Servlet 技术实现，主要包括了之前后端系统划分当中提到的各个后端处理，例如登录功能的具体实现、新闻管理、通知管理、资源下载管理的一个具体的实现等等。

数据访问层负责与数据库的交互，实现数据的持久化操作。该层通过 JDBC 技术实现，并与各个 Servlet 进行联动，以确保各个 Servlet 的功能可以在数据库中进行永久化保存等。

表示层通过调用业务逻辑层提供的接口获取数据和处理结果，业务逻辑层负责处理具体的业务规则，并通过数据访问层进行数据操作，数据访问层直接与数据库交互，为上层提供数据服务。三层之间相互调用形成依赖关系。

逻辑架构从整体上说明了系统的各个层次，梳理了模块之间的关系，确定了基本的技术选型，是后续开发、部署工作的蓝图，对系统实现有重要的指导意义。

4.2 系统功能设计

针对需求分析划分的模块，一一对它们的功能进行具体说明。

4.2.1 登录模块：

正如之前所展示的，登录模块分为了验证码模块和登录验证模块。其中，验证码模块又包括了生成验证码和检测验证码。用户在登录界面打开就会自动产生一个验证码，并需要用户输入。输入后先进行验证码的校验，合格才进行账号密码的检验。

4.2.2 新闻动态模块：

新闻动态模块也分为了两个部分：新闻和通知。新闻动态当中则包括了新闻管理模块，其中可以实现新闻发布、新闻预览、新闻管理。同时，出于简化新闻预览功能和更方便的管理新闻，实现了过渡的草稿功能，即新闻需要先保存到新闻草稿当中，方便进行预览确保合格之后再发布出去。同时，针对新闻展示界面也做出了自己的实现。同时，也按照要求实现了新闻的评论功能。

通知部分主要实现了通知的一个发布和删除。通知分为教务通知和通知公告两个部分，管理界面可以同时管理两种通知。展示界面则和其他界面相似。

4.2.3 公共服务模块：

公共服务模块在动态方面主要实现的是资源管理。公共服务需要实现一个“资源下载”的功能。这就需要实现管理界面，方便管理员实现资源的上传和下载。同时参照学校官网，资源分为了学生相关、教务相关、科研相关、基础课程四个种类。

4.2.4 静态页面模块：

静态页面的覆盖面则较广，覆盖了从学院概括、学科建设到本科教学、研究生以及公共服务板块当中的部分界面。

4.3 系统数据库设计

数据库设计其实和上述的系统功能设计是息息相关的，因为数据库的永久化保存主要实现的就是动态界面当中需要的内容。根据系统分析和业务模块的划分结果，针对结构化数据，系统采用 SQL Server 完成保存和处理。

接下来将分别讲述各个数据库表设计。

登录表实现：

表 1 登录表

字段	类型	约束	说明
userId	char(5)	primary key	用户名
uPassword	nvarchar(100)	not null	用户密码
identity	varchar(20)	只能为 administrator、user	用户身份

出于简化目的，这里用户 id 只设置了 5 位。用户密码设置了 nvarchar(100) 是因为密码会使用实现的一个 MD5 加密，所以具体的位数会较大一点。用户身份则只有两个可以实现：administrator 和 user，代表管理员和普通用户。

新闻表实现：

表 2 新闻表

字段	类型	约束	说明
id	char(32)	primary key	新闻的 id
title	varchar(100)	not null	新闻标题
createTime	date	not null	新闻创建时间
content	varchar(max)		新闻正文内容
creator	varchar(100)		新闻供稿者

这里参照学院官网具有的一个新闻的实现模式，设置了包括供稿者在内的 5 个字段。新闻的 id 则是出于简化目的，通过对 title+createTime 进行 hash 得到。

当然，正如之前讲述的，这里也有临时新闻表来存储新闻草稿。临时新闻表的结构和新闻表的一致，如下：

表 3 临时新闻表

字段	类型	约束	说明
id	char(32)	primary key	新闻的 id
title	varchar(100)	not null	新闻标题
createTime	date	not null	新闻创建时间
content	varchar(max)		新闻正文内容
creator	varchar(100)		新闻供稿者

新闻评论表实现：

表 4 新闻评论表

字段	类型	约束	说明
id	char(32)	primary key	评论的 id
newsid	char(32)	外键到新闻表	新闻的 id
userId	char(5)	外键到用户表	评论者的 id
comment	varchar(200)		评论内容
createTime	date	not null	发布时间

同时，为了实现新闻的评论功能，自然也需要实现新闻评论表。在评论表中设置了评论的 id（方便后续拓展实现回复功能）、新闻 id、用户 id、评论内容和创建时间 5 个字段，方便评论区的一个具体实现。

通知表实现：

表 5 通知表

字段	类型	约束	说明
noticeId	varchar(32)	primary key	通知 id
title	nvarchar(255)	not null	通知标题
publishTime	DATE	not null	通知发布时间
publishKinds	varchar(20)	只能为 academic、notice	通知种类
content	nvarchar(MAX)		通知正文

针对通知的一个具体实现，也是参照了学院的各个通知总结出来的一个字段内容。与新闻不同的是，通知一般不需要供稿者，所以这里选择了删除掉。针对不同种类的通知，这里也在 kinds 当中设置了两值，分别为 academic 代表教务通知和 notice 代表通知公告。

同时，针对通知当中各个文件的地址附件链接，由于可能会存在一个通知有 0 个、1 个到多个链接的可能性，于是单独开设一张表来存储通知的链接：

表 6 通知链接表

字段	类型	约束	说明
noticeId	varchar(32)	外键到通知表	通知 id
filePath	nvarchar(500)	not null	文件路径

针对资源下载功能，实现的表为资源表和资源链接表。其实具体内容和通知是十分相似的，如下：

表 7 资源表

字段	类型	约束	说明
resourceId	varchar(32)	primary key	资源 id
title	nvarchar(255)	not null	资源标题
publishTime	DATE	not null	资源发布时间
publishKinds	varchar(20)	只能为 student、teaching、research、basic	资源种类

其中，资源表自然不需要正文字段，于是进行了删除。种类字段则是实现了 student（学生相关）、teaching（教务相关）、research（科研相关）、basic（基础课程）四个部分，与实际的学院官网是相同的。

同样的，针对资源当中各个文件的地址附件链接，由于可能会存在有 0 个、1 个到多个链接的可能性，于是单独开设一张表来存储资源具体的链接：

表 8 资源链接表

字段	类型	约束	说明
resourceId	varchar(32)	外键到资源表	资源 id
filePath	nvarchar(500)	not null	文件路径

综上，给出了所有具体的一个数据库实现。

4.4 本章小结

本章对计算机学院门户网站系统进行了详细的系统设计，从总体架构、功能模块到数据库设计进行了全面的阐述。

在系统总体设计方面，采用了基于 B/S 架构的三层结构设计模式。通过表示层（JSP）、业务逻辑层（Servlet）和数据访问层（JDBC）的分层设计，明确了系统各层次的职责和相互关系，为后续开发工作提供了清晰的技术路线。

通过本章的系统设计，为系统的具体实现提供了完整的技术方案和数据结构支持，确保了后续开发工作的顺利进行。这种设计既保证了系统功能的完整性，又为未来的功能扩展预留了空间。

5 系统实现与测试

5.1 系统环境

系统环境是系统开发和运行的基础，在正式开发前，需要搭建好基础环境。本系统的主要任务是实现一个计算机学院的门户网站，因此系统环境的侧重点在网页开发。

表 9 主要环境表

名称	版本号	说明
IntelliJ IDEA	2023.3.2	集成开发工具
JDK	21.0.2	Java 开发与运行环境
SQL Server	2022- 16.0.1135.2	关系数据库
JDBC	12.6.2	数据库连接包
Tomcat	11.0.0	Web 应用服务器

5.2 关键功能实现

接下来将分别讲述各个细分后的模块的实现：

(1) 加密模块：

```
public static String getHash(String str){
    MessageDigest hash = null;
    try{
        hash = MessageDigest.getInstance( algorithm: "MD5");
        byte[] bytes = hash.digest(str.getBytes(StandardCharsets.UTF_8));
        StringBuilder result = new StringBuilder();
        for (byte b : bytes) {
            String temp = Integer.toHexString( i: b & 0xff);
            if (temp.length() == 1) {
                result.append("0").append(temp);
            } else {
                result.append(temp);
            }
            //result.append(temp);
        }
        return result.toString();
    } catch (NoSuchAlgorithmException e) {
        System.out.println(e.getMessage());
    }
    return "";
}
```

图 2 加密算法

通过这个得到加密后的值，实现了许多功能，包括且不限于密码加密等，是项目的基础功能之一。

(2) 登录过滤模块：

通过写一个 **filter** 过滤器，实现在未登录的情况下无法访问某些网页的功能。

```
@WebFilter({"newsTrends/academicNotice.jsp","newsTrends/newsManagement.jsp",
"/newsTrends/noticeList.jsp","publicService/*","newsTrends/noticeManagement.jsp"})
```

图 3 过滤器配置

通过注解实现了过滤器所需要过滤的一些网页的配置。具体实现也较为简单，如下：

```
@Override
public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain) throws IOException, ServletException {
    HttpServletRequest req = (HttpServletRequest) request;
    HttpServletResponse res = (HttpServletResponse) response;

    // 检查用户是否已经登录
    HttpSession session = req.getSession(b: false);
    if (session != null && session.getAttribute(s: "identity") != null) {
        // 用户已登录，继续处理请求
        chain.doFilter(request, response);
    } else {
        // 用户未登录，重定向到登录页面
        res.sendRedirect( location: req.getContextPath() + "/login.jsp");
    }
}
```

图 4 过滤实现

不难看出，只需要通过检测 **session** 当中是否有 **identity** 字段就可以检测是否成功登录了。如果没有，就会将页面请求重定向到登录界面。

(3) 验证码模块：

```
//在内存当中创建图像
int width=60,height=40;

BufferedImage image=new
BufferedImage(width,height,BufferedImage.TYPE_INT_RGB);

//获取图形上下文
Graphics g=image.getGraphics();

Random random=new Random();

//设定背景色
```



```

g.setColor(new Color(237, 236, 233));
g.fillRect(0,0,width,height);
g.setFont(new Font("Times New Roman",Font.PLAIN,18));
//随机产生干扰线
for(int i=0;i<200;++i){
    int x=random.nextInt(width);
    int y=random.nextInt(height);
    int x1=random.nextInt(12);
    int y1=random.nextInt(12);
    g.drawLine(x,y,x1,y1);
}
//获取随机产生的验证码（4 位数字）
StringBuilder sRand= new StringBuilder();
for(int i=0;i<4;i++){
    String rand=String.valueOf(random.nextInt(10));
    sRand.append(rand);
    g.setColor(new
Color(30+random.nextInt(160),40+random.nextInt(170),40+random.nextInt(180)));
    g.drawString(rand,13*i+6,16);
}
request.getSession().setAttribute("verificationCode",sRand.toString());
g.dispose();
ImageIO.write(image,"JPEG",response.getOutputStream());

```

通过参照课本上的代码，实现了一个 4 位数字验证码的生成，并将生成的验证码保存到 session 当中便于后续检验，并将生成的图片返回给所需要的登录界面。

(4) 验证码验证模块：

```
String verificationCode = request.getParameter( s: "verificationCode");
```

图 5 request 获取输入的验证码

```
// 从Session中获取验证码进行比较  
HttpSession session = request.getSession();  
String sessionCode = (session != null) ? (String) session.getAttribute( s: "verificationCode") : null;
```

图 6 session 获取生成的验证码

随后使用 `equal()` 进行检测是否相等即可。如果相等则重定向到登录验证模块。

```
// 向Servlet2发送请求以验证账号密码  
String servletUrl = request.getContextPath() + "/loginServlet";  
// 重定向到Servlet2  
response.sendRedirect(servletUrl);
```

图 7 重定向语句

如果失败则返回一个 js 指令，来提示验证码错误：

```
out.println("<script type='text/javascript'>");  
out.println("alert('验证码错误');");  
out.println("window.history.back();");  
out.println("</script>");
```

图 8 返回错误信息

(5) 登录验证模块：

```
// 调用loginDao.query方法验证账号密码  
boolean isValid = loginDao.query(userId, userPassword);
```

图 9 登录验证账户密码

获取账户密码的方式较为简单，这里不再展示。根据图 6 可以看出，在获得账户密码之后，会调用 DAO 层的函数去查验是否符合数据库当中的记录。关于具体的 DAO 层实现后续讲述。

```

out.println("<script type='text/javascript'>");
out.println("alert('登录成功! ');");
out.println("window.location.href='" + request.getContextPath() + "/login.jsp'");
out.println("</script>");

```

图 10 登录成功提示

```

out.println("<script type='text/javascript'>");
out.println("alert('账号或密码错误');");
out.println("window.location.href='" + request.getContextPath() + "/login.jsp'");
out.println("</script>");

```

图 11 登录失败

同样的，也在 response 当中写入了 js 指令来提示用户登录是否成功，起到了一个良好的用户交互。

(6) 新闻动态管理：

新闻动态管理的功能较为复杂，下面将分别讲述。

① 新闻发布功能：

新闻发布功能需要实现的功能难点主要在于图片的上传。

很显然，新闻是一个需要图片展示的事物。图片在新闻当中是至关重要的一环，那么实现图片的一个保存就十分重要。该如何实现多张图片的上传保存呢？同时还需要将图片插入到正文当中。

在这里实现了一个 JavaScript 函数进行实现。我们先来查看具体的前端代码界面：

```

<form class="news-form" action="" method="post" enctype="application/x-www-form-urlencoded">
  <label for="title">标题:</label>
  <input type="text" id="title" name="title" required>
  <label for="publish-time">发布时间:</label>
  <input type="date" id="publish-time" name="publish-time" required>
  <label for="author">供稿人:</label>
  <input type="text" id="author" name="author" required>
  <label for="content">内容:</label>
  <textarea id="content" name="content" rows="10" required></textarea>
  <button type="button" class="insert-image">插入图片</button>
  <input type="submit" id="news-submit" value="保存">
</form>

```

图 12 新闻发布表单

```

<form action="../imageUploadServlet" method="post" enctype="multipart/form-data">
  <!-- 隐藏的file用于文件上传 -->
  <input type="hidden" id="none-title" name="title">
  <input type="file" id="image-input" name="image" accept="image/*" style="...">
  <input type="submit" id="image-submit" value="提交" style="...">
</form>

```

图 13 图片上传表单

可以看出其实图片上传的表单和新闻发布的表单并不是同一个表单。其具体实现依赖于下列 JavaScript 函数：

```

// 触发文件选择器
document.querySelector(selectors: '.insert-image').addEventListener( type: 'click', listener: function() : void {
  document.getElementById( elementId: 'image-input').click();
});

```

图 14 触发图片上传功能按钮

可以看出，通过这个 js 函数，实现了点击新闻发布表单当中的图片上传按钮之后，图片上传表单就会被点击一次，实现一次图片选择。

```

//选择完图片之后，上传图片到本地的临时文件夹中
document.getElementById('image-input').addEventListener('change',
function(event) {
  const file = event.target.files[0];
  if (file) {
    const formData = new FormData();
    formData.append('image', file);
    formData.append('title', document.getElementById('title').value);

    formData.append('date',document.getElementById('publish-time').value)

    const xhr = new XMLHttpRequest();
    xhr.open('POST', '../imageUploadServlet', true);
    xhr.onload = function() {
      if (xhr.status === 200) {

```

```

        const response = JSON.parse(xhr.responseText);
        const filename = response.fileName;
        if (response.success) {
            const newImageTag = `<img alt=""
src="" + filename + ">`; // 使用服务器返回的文件名
            const contentArea =
document.getElementById('content');
            //const currentContent = contentArea.value;
            contentArea.value += newImageTag; // 在内容末尾添
加新的<img>标签
        } else {
            alert('上传失败: ' + response.message);
        }
    } else {
        alert('上传失败: ' + xhr.responseText);
    }
};
xhr.send(formData);
}
});

```

通过上述函数可以得到其具体实现的功能是在用户选择图片之后，将图片获取并组成一个 FormData 的形式，并通过一个自定义的 Ajax 请求发送给对应的 Servlet，并将 Servlet 返回的文件名打上标签后插入到新闻正文的 textarea 中。

```

@Override
protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    Part part = request.getPart("image");
    //获取文件名
    String photo = part.getSubmittedFileName();
}

```

```

//获取标题和日期

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

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

if(!(photo.endsWith(".jpg")||photo.endsWith(".png")||photo.endsWith(".jpeg"))){

    request.setAttribute("type","err");

    try {

        //如果格式不对，跳转

        response.sendRedirect(request.getContextPath()

+
"/newsTrends/newsManagement.JSP");

        } catch (IOException e) {

            System.out.println(e.getMessage());

        }

        photo="";

    }

    photo=hash.hash.getHash(title+date)+photo;

    //本地目录

    String path="F:\\

    大

    学

\\JavaEE\\code\\final_experiment\\web\\temp\\img";

    File file = new File(path);

    if(!file.exists()){

        //如果目录不存在就新建一个

        file.mkdirs();

    }

    try {

        //本地名字+文件名字 将文件的名称写入本地

        part.write(path+"/"+photo);

    } catch (IOException e) {

        System.out.println(e.getMessage());

```

```

    }

    if(photo.isEmpty()){
        return;
    }

    PrintWriter out = response.getWriter();

    String jsonBuilder = "{" +
        "\"success\": true," +
        "\"fileName\": \"" + photo + "\"" +
        "}";

    out.print(jsonBuilder);

    out.flush();
}

```

通过上述 Servlet 当中实现的函数,可以得出 Servlet 实现的功能为接收 request 请求后,会通过 part 类来获取 FormData 当中保存的图片,并在一系列检查之后,通过 File 类创建目录(如果没有目录)和 Part.write()函数将图片写入到指定的地址”F:\大学\JavaEE\code\final_experiment\web\temp\img”当中。

同时,在最后自定义地组成了 json 格式的回复内容:



```

String jsonBuilder = "{" +
    "\"success\": true," +
    "\"fileName\": \"" + photo + "\"" +
    "}";

```

图 15 返回内容

以此告诉 JavaScript 函数插入成功了,并将文件名返回。

接下来新闻的 form 当中就只有一些文字内容了,具体的上传实现极为容易,这里不过多展开,只给出代码:

```

event.preventDefault(); // 阻止表单的默认提交行为
const title = document.getElementById( 'title').value;
const publishTime = document.getElementById( 'publish-time').value;
const author = document.getElementById( 'author').value;
let content = document.getElementById( 'content').value;
// 将转义的换行符替换为非转义的换行符
content = content.replace(/\\n/g, '\\n');
// 发送请求
const xhr : XMLHttpRequest = new XMLHttpRequest();
xhr.open( method: 'POST', url: '../newsTempUploadServlet', async: true);
xhr.setRequestHeader( name: 'Content-Type', value: 'application/x-www-form-urlencoded');
xhr.onload = function () : void {
    if (xhr.status === 200) {
        const response = JSON.parse(xhr.responseText);
        console.log(xhr.responseText);
        if (response.success) {
            alert('上传成功');
            // 清除表单
            document.getElementById( 'title').value = '';
            document.getElementById( 'publish-time').value = '';
            document.getElementById( 'author').value = '';
            document.getElementById( 'content').value = '';
        } else {
            alert('上传失败: ' + response.message);
        }
    } else {
        alert('上传失败: ' + xhr.responseText);
    }
};
// 直接以 URL 编码的形式发送数据
console.log(publishTime);
xhr.send( body: `title=${title}&publishTime=${publishTime}&author=${author}&content=${content}`);

```

图 16 新闻发布代码

但是，本系统当中新闻发布不会直接插入到新闻数据库当中，而是会插入到临时新闻数据库，即新闻草稿当中，便于管理员进一步的确认新闻书写是否成功。

② 新闻草稿管理：

新闻草稿界面的内容需要动态加载，于是在打开（点击对应的按钮）时会使用 JavaScript 函数自动的发送请求给对应的 Servlet 来获取数据：

```

//获取新闻草稿当中的内容，动态生成
const xhr = new XMLHttpRequest();
xhr.open('GET', '../newsTempUploadServlet', true);
xhr.onload = function() {

```



```

if (xhr.status === 200) {
    // console.log(xhr.responseText);
    const data = JSON.parse(xhr.responseText);
    const table = document.createElement('table');
    const thead = table.createTHead();
    const headerRow = thead.insertRow();
    const headers = ['标题', '发布时间', '供稿人', '行为'];
    for (let header of headers) {
        const th = document.createElement('th');
        th.textContent = header;
        headerRow.appendChild(th);
    }
    const tbody = table.createTBody();
    data.forEach(item => {
        //组成表格， 略
        const actionsCell = row.insertCell(3);
        const publishLink = document.createElement('a');
        publishLink.textContent = '发布';
        publishLink.href = 'javascript:void(0)';
        publishLink.addEventListener('click', function () {
            //略
        });
        actionsCell.appendChild(publishLink);
        const previewLink = document.createElement('a');
        previewLink.textContent = '预览';
        previewLink.href = 'javascript:void(0)';
        previewLink.addEventListener('click', function () {
            //略
        });
    });
}

```

```

        actionsCell.appendChild(previewLink);

        const deleteLink = document.createElement('a');
        deleteLink.textContent = '删除';
        deleteLink.href = 'javascript::void(0)';
        deleteLink.addEventListener('click', function () {
            //略
        });
        actionsCell.appendChild(deleteLink);
    });
    document.querySelector('.newsDraft').appendChild(table);
    // console.log('生成表格成功');
} else {
    alert('获取新闻列表失败:'+ xhr.responseText);
}
};
xhr.send();

```

上述函数是简化后的一个 js 函数，简单来讲就是根据返回的一个数组，进行一个 for 循环来自动生成一个 table 的每一项，并在每一项的后续都附上发布、预览、删除三个按钮。由于三个功能的实现较为简单，都是简单的发送一个 Ajax 请求并在对应的 Servlet 当中调用数据库，这里不过多展示。下面讲述获取新闻草稿的 Servlet 的主要实现：

```

@Override
protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws IOException {
    // 设置请求的编码格式
    request.setCharacterEncoding("UTF-8");
    response.setContentType("application/json;charset=UTF-8");
    response.setCharacterEncoding("UTF-8");
    // 获取新闻数据

```

```

        ArrayList<Object[]>                newsList                =
Dao.newsTrends.newsTempUploadDao.getNews();

        // 将 Object[] 转换为 String[]
        ArrayList<String[]> stringNewsList = new ArrayList<>();
        for (Object[] objArray : newsList) {
            String[] stringArray = new String[objArray.length];
            for (int i = 0; i < objArray.length; i++) {
                stringArray[i] = objArray[i] != null? objArray[i].toString() :
"";

            }
            stringNewsList.add(stringArray);
        }

        // 手动构建 JSON 格式的字符串
        StringBuilder jsonBuilder = new StringBuilder("[");
        for (String[] news : stringNewsList) {
            jsonBuilder.append("{\"title\":\"").append(news[0]).append("\",");

jsonBuilder.append("\"createTime\":\"").append(news[1]).append("\",");

jsonBuilder.append("\"creator\":\"").append(news[2]).append("\"},");
        }
        if (jsonBuilder.length() > 1) {
            jsonBuilder.setLength(jsonBuilder.length() - 1);
        }
        jsonBuilder.append("]");

        // 将 JSON 数据写入响应

```

```
        PrintWriter out = response.getWriter();
        out.print(jsonBuilder);
    }
```

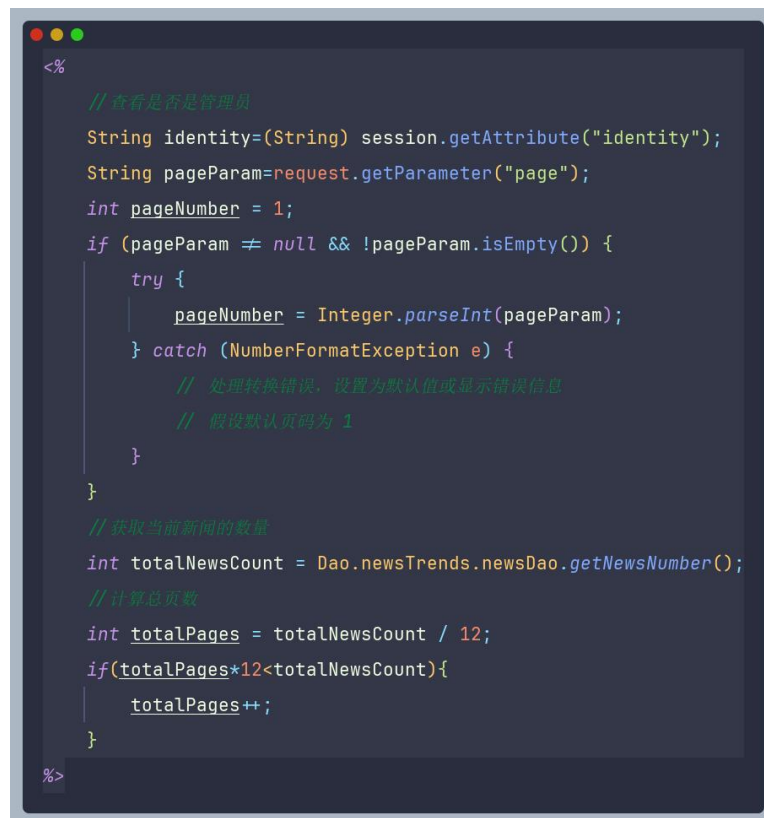
这个 Get 请求处理函数实际逻辑也很简单：从数据库当中获取数据、数据格式转化、数据组成 JSON。这里不过多讲述。

③ 新闻管理：

新闻管理的实现与新闻草稿十分相似，无非是调用了不同的数据库而已，这里不过多讲述。

(7) 新闻动态展示：

首先，针对新闻必然数目庞大的问题，在新闻总体的一个展示页面就必须要实现分页功能。



```
<%
    // 查看是否是管理员
    String identity=(String) session.getAttribute("identity");
    String pageParam=request.getParameter("page");
    int pageNumber = 1;
    if (pageParam != null && !pageParam.isEmpty()) {
        try {
            pageNumber = Integer.parseInt(pageParam);
        } catch (NumberFormatException e) {
            // 处理转换错误，设置为默认值或显示错误信息
            // 假设默认页码为 1
        }
    }

    // 获取当前新闻的数量
    int totalNewsCount = Dao.newsTrends.newsDao.getNewsNumber();
    // 计算总页数
    int totalPages = totalNewsCount / 12;
    if(totalPages*12<totalNewsCount){
        totalPages++;
    }
%>
```

图 17 java 脚本

这段 java 代码的作用主要是：1.获取 session 当中关于用户的身份信息 2.获取当前的页码 3.获取总体新闻的数量并计算总页数(设置了一页最多 12 个新闻)

```
<nav aria-label="Page navigation example">
    <ul class="pagination">
```

```

        <li class="page-item" <% if(pageNumber == 1)
{ out.print("disabled"); } %>">
        <a class="page-link" href="newsTrends.JSP?page=<%=
pageNumber-1%>" aria-label="Previous">
            <span aria-hidden="true">&laquo;</span>
        </a>
    </li>
    <%
    if(totalPages<5){
        for(int i=1;i<=totalPages;i++) {
            out.print("<li class='page-item'><a
class='page-link' href='\"newsTrends.JSP?page=\" + i + \">\" + i + \"</a></li>");
        }
    }
    else{
        if(pageNumber<3){
            for(int i=1;i<pageNumber+2;++i){
                out.print("<li class='page-item'><a
class='page-link' href='\"newsTrends.JSP?page=\" + i + \">\" + i + \"</a></li>");
            }
        }
        else{
            out.print("<li class='page-item'><a
class='page-link' href='\"newsTrends.JSP?page=1\">1</a></li>");
            out.print("<li class='page-item disabled'><a
class='page-link' href='\"#\">...</a></li>");
            for(int
i=pageNumber-2;i<pageNumber+2&&i<totalPages;++i){
                out.print("<li class='page-item'><a

```

```

class="page-link" href="newsTrends.JSP?page=" + i + ">" + i + "</a></li>");
    }

    if(pageNumber+2<totalPages){
        out.print("<li        class=\"page-item
disabled\"><a class=\"page-link\" href=\"#">...</a></li>");
        out.print("<li        class=\"page-item\"><a
class=\"page-link\"
href=\"newsTrends.JSP?page="+totalPages+"\">" + totalPages + "</a></li>");
    }
}
}

%>
<li class="page-item <% if(pageNumber == totalPages)
{ out.print("disabled"); } %>">
    <a class="page-link" href="newsTrends.JSP?page=<%=
pageNumber+1%>" aria-label="Next">
        <span aria-hidden="true">&raquo;</span>
    </a>
</li>
</ul>
</nav>

```

上面的 html 代码夹杂 java 代码看似复杂，实际上是实现了页码的功能。简单梳理为：前一页（当前页码为 1 时不可用）、生成页面按钮（如果总页数小于 5 则生成全部的按钮，否则查看当前页码是否小于 3。如果小于 3 则生成 1 2 3 4 5 五个页码按钮，否则生成 1 ... page-2 page-1 page ... totalPages）、后一页（页码为最后一页时不可用）。以此实现更方便用户的一个页码选择方式。

同时，针对具体的一个新闻展示，这里采取了 **JavaBean** 的实现方式。在获取新闻的时候，是通过查询 id 来实现的，那么我们就可以定义一个类 **news**：

```

String id,title,content,createTime,author;

public news() {
}

public news(String id) {
    this.id = id;
    // 获取新闻内容, 生成html文件
    Object[] news =Dao.newsTrends.newsDao.getNews(id);
    this.title=news[1].toString();
    this.createTime=news[2].toString();
    this.content=news[3].toString();
    this.author=news[4].toString();
}

public String getId() { return id; }
public void setId(String id) {
    this.id = id;
    // 获取新闻内容, 生成html文件
    Object[] news =Dao.newsTrends.newsDao.getNews(id);
    this.title=news[1].toString();
    this.createTime=news[2].toString();
    this.content=news[3].toString();
    this.author=news[4].toString();
}

```

图 18 news 类

从图 18 当中可以看出 news 类的一些属性和 setId 函数的功能为设置 id 后从数据库当中获取新闻数据并保存到 news 类当中。当然，后续还有一系列其他的 getter 和 setter。

同时，也在 news 类当中实现了获取评论的功能。主要逻辑是从数据库当中获取到评论数据并将其转化为对应的 html 格式，同时还会根据函数输入参数 isadmin 来确认是否需要在评论的后面添加一个删除按钮。

具体代码如下图：

```

// 获取评论, 生成html文件
1 个用法
//
public String getComments(boolean isAdmin){
    ArrayList<Object[]> comments=Dao.newsTrends.newsCommentsDao.getComments(this.id);
    StringBuilder commentHtml= new StringBuilder("<div>");
    for (Object[] comment:comments){
        String comment_ = comment[1].toString();
        int index=0;
        while ((index=comment_.indexOf( str: "\\n", index))!=-1){
            comment_=comment_.replace( target: "\\n", replacement: "\\n");
        }
        commentHtml.append("<div>");
        commentHtml.append("<span class=\"userId\">").append(comment[0].toString()).append("</span>");
        commentHtml.append("<span class=\"comment\">").append(comment_).append("</span>");
        commentHtml.append("<span class=\"time\">").append(comment[2].toString()).append("</span>");
        if (isAdmin) {
            commentHtml.append("<button class=\"delete-btn\" onclick='deleteComment(\"')'.append(comment[3].toString())
                .append(\"')>删除</button>");
        }
        commentHtml.append("</div>");
    }
    commentHtml.append("</div>");
    return commentHtml.toString();
}

```

图 19 获取评论

从中可以看出删除按钮的函数是 deleteComment(), 具体实现如下:

```

function deleteComment(commentId) {
    if (!confirm("确定要删除这条评论吗? ")) {
        return; // 如果用户点击取消, 则不执行删除
    }
    const xhr = new XMLHttpRequest();
    xhr.open("POST", "../newsCommentDeleteServlet", true);
    xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
    xhr.onload = function () {
        if (xhr.status === 200) {
            const response = xhr.responseText;
            alert(response);
            if (response === "删除成功") {
                // 如果删除成功, 可以在这里刷新本界面
                location.reload();
                /*const commentElement = document.querySelector('.comment-item[data-comment-id="'+ commentId + '"]');
                if (commentElement) {
                    commentElement.parentNode.removeChild(commentElement);
                }*/
            }
        }
    };
    xhr.send("commentId=" + encodeURIComponent(commentId));
}

```

图 20 删除评论函数

实现逻辑也很简单，给用户（管理员）一个确认是否删除的选择框，用户选择确认之后定义一个 Ajax 的请求并发送给对应的 Servlet。Servlet 具体实现较为简单，这里不过多展示。

那么在我具体的 news.JSP 当中的实现就更为简单了，如下：



```
<head>
<%
    String id=request.getParameter("id");
    String path=request.getContextPath();
%>
<jsp:useBean id="currentNews" class="newsTrends.news" scope="page">
    <jsp:setProperty name="currentNews" property="id" value="<%=id%>" />
</jsp:useBean>
<title><jsp:getProperty name="currentNews" property="title" /></title>
</head>
```

图 21 使用 JavaBean

在 head 当中通过 request 获取 id 后，就可以使用 JavaBean 初始化一个 news 类，并在后续对其进行具体的使用。例如针对新闻的正文内容，就可以只用一个函数调用带过：



```
<%
    out.write(currentNews.getNewsContent(path));
%>
```

图 22 获取新闻正文内容

其中，getNewsContent()是在 news 类当中定义的，由于逻辑简单代码又较多，这里不过多展示。

(8) 新闻评论功能：

建立在 news 类的基础上，只需要使用 JavaBean+java+html 就可以很简单的实现：

```

<section id="commentsSection">
    <%
        // 检查用户是否已登录
        boolean isAdmin=false;
        String userId = (String) session.getAttribute("userId");
        String identity = (String) session.getAttribute("identity");
        if (identity == null) {
            // 用户未登录，显示登录提示
        }
    %>
    <p>请<a href="${pageContext.request.contextPath}/login.jsp">登录</a>后评论。</p>
    <%
    } else {
        // 用户已登录，设置管理员+显示评论表单
        isAdmin = "administrator".equals(identity);
    %>
    <form action=" ../newsCommentServlet" method="get">
        <input type="hidden" name="newsId" value="<jsp:getProperty name="currentNews" property="id" />" />
        <input type="hidden" name="userId" value="<%= userId %>" />
        <label for="commentContent">评论内容:</label>
        <textarea id="commentContent" name="commentContent" maxlength="200" placeholder="善语结善缘，恶语伤人心" required></textarea>
        <button type="submit" id="comment-submit">提交评论</button>
    </form>
    <%
    }
    %>
    <%
        out.write(currentNews.getComments(isAdmin));
    %>
</section>

```

图 23 评论区实现

可以看出逻辑也很简单。判断用户是否登录，如果登录了就可以出现发送评论的 form，如果未登录则会提示登录后可评论。同时，调用先前展示的 getComments()函数来展示评论区的具体内容。

(9) 通知公告管理：

通知公告管理和新闻发布的管理十分相似，是其简化版。通知公告不需要新闻那么的“小心”，所以只需要实现正常的直接发布即可，不需要实现草稿功能。其发布表单如下：

```

<form class="notice-form" action="../noticeServlet" method="post" enctype="multipart/form-data">
  <label for="title">标题:</label>
  <input type="text" id="title" name="title" required>
  <label for="publish-time">发布时间:</label>
  <input type="date" id="publish-time" name="publish-time" required>
  <label for="publish-kinds">资源种类</label>
  <select id="publish-kinds" name="publish-kinds" required>
    <option value="academic">教务通知</option>
    <option value="notice">通知公告</option>
  </select>
  <label for="content">内容:</label>
  <textarea id="content" name="content" rows="10" required></textarea>
  <label for="file-inputs-container">附件:</label>
  <div id="file-inputs-container">
    <div class="file-input-group">
      <input type="file" class="file-input" name="file[]" accept="*">
    </div>
  </div>

  <input type="submit" id="news-submit" value="保存">
</form>

```

图 24 通知发布表单

可以看出通知公告的发布类别是 **academic** 教务通知和 **notice** 通知公告两类。符合我们的定义。同时，由于一个公告可能会伴随着多个文件，所以针对一个 **input** 必然是不够的，所有实现如下内容：

```

const container : HTMLElement = document.getElementById( 'file-inputs-container' );
// 监听container内的所有file input的change事件
container.addEventListener( type: 'change', listener: function( e : Event ) : void {
  if ( e.target.classList.contains( 'file-input' ) ) {
    // 检查是否是最后一个input
    const inputs : HTMLCollectionOf<Element> = container.getElementsByClassName( className: 'file-input' );
    const currentInput : EventTarget = e.target;

    if ( currentInput === inputs[inputs.length - 1] ) {
      // 创建新的input组
      const newGroup : HTMLDivElement = document.createElement( tagName: 'div' );
      newGroup.className = 'file-input-group';

      const newInput : HTMLInputElement = document.createElement( tagName: 'input' );
      newInput.type = 'file';
      newInput.className = 'file-input';
      newInput.name = 'file[]';
      newInput.accept = '*';

      newGroup.appendChild( newInput );
      container.appendChild( newGroup );
    }
  }
});

```

图 25 JavaScript 函数

函数的功能是当最后一个 File input 改变（即选择了一个文件）时，会在后面自动生成一个新的 File input，便于用户选择多个文件。

剩余的具体管理部分和新闻相似，从数据库中获取全部的数据，并生成对应的 table。并且出于简化目的，后续只需要实现一个删除功能即可，不需要实现预览功能。

(10)通知公告展示：

与新闻的十分相似，定义一个 notice 类并通过 JavaBean 最终实现。这里展示部分 notice 类的代码和部分前端 JSP 文件的代码：

```
private String id,title,publishTime,content;
4 个用法
private List<Object[]> links;
1 个用法
private static final Map<String, String> NOTICE_PATHS = new HashMap<>() {{
    put("academic", "/static/notice/academicNotice");
    put("notice", "/static/notice/noticeList");
}};

9 个用法
public notice() {
}

9 个用法
public notice(String id) {
    this.id = id;
    // 获取通知信息
    List<Object[]> notices = noticeDao.getNoticesById(id);
    if (!notices.isEmpty()) {
        Object[] notice = notices.getFirst();
        this.title = notice[0].toString();
        this.publishTime = notice[1].toString();
        this.content = notice[2].toString();
    }
    // 获取通知附件链接
    this.links = noticeDao.getNoticeLink(id);
}

public String getId() { return id; }
public void setId(String id) {
    this.id = id;
    // 获取通知信息
    List<Object[]> notices = noticeDao.getNoticesById(id);
    if (!notices.isEmpty()) {
        Object[] notice = notices.getFirst();
        this.title = notice[0].toString();
        this.publishTime = notice[1].toString();
        this.content = notice[2].toString();
    }
    // 获取通知附件链接
    this.links = noticeDao.getNoticeLink(id);
}
```

图 26 notice 类



图 27 JSP 初始化 notice 类

```

<div style="margin-top:30px">
    <h4 style="margin-bottom: 10px;text-align: center;font-size:
22px;font-family: '微软雅黑',serif;">
        <JSP:getProperty name="currentNotice" property="title" />
    </h4>
    <h4 style="margin-bottom: 10px;text-align:
center;font-size:14px;font-family: '微软雅黑',serif;color:darkgray">
        发布日期:
        <JSP:getProperty name="currentNotice" property="publishTime" />
    </h4>
    <div>
        <!-- 这里是通知正文内容 -->
        <div style="margin: 20px 0;">
            <%= currentNotice.getHtmlContent()%>
        </div>

        <!-- 这里是附件下载部分 -->
        <%= List<Object[]> links = currentNotice.getLinks();
            if (links == null || links.isEmpty()) {
        %>
        <p>暂无附件</p>
    
```

```

        <% } else { %>

        <div style="margin-top: 20px;">

            <h6>附件: </h6>

            <%    for(Object[]    link    :    links){    String
filePath=link[0].toString(); String
                fileName=filePath.substring(32); // 更灵活的文件
名获取方式 %>

                <p>

                    <span

style="box-sizing:border-box;font-size:20px;font-family:    微    软    雅    黑    ,
sans-serif;color:#212529;">

                        <a                                href="<%=
currentNotice.getNoticePath(type, path) + " /" + filePath %>"

style="box-sizing:border-box;font-size:16px;">

                            <span

style="box-sizing:border-box;color:#212529;">

                                <%= fileName %>

                            </span>

                        </a>

                    </span>

                </p>

            <% } %>

        </div>

    <% } %>

    <p><br></p>

</div>

</div>

```

上述代码的主要功能是将 **resource** 类当中的内容转化为对应的 **html** 格式并进行展示。与之前的新闻的区别在于新闻的转化是在 **news** 类的函数当中的，这个 **html** 的展示是部分放置在 **JSP** 文件当中的。

剩余部分不做展示。

(11) 资源下载管理：

资源下载管理相较于通知公告管理更为简单，它只需要将通知公告 **form** 当中的 **content** 栏进行删除，并在数据库操作上进行一定的修改即可实现。这里不进行展示。

(12) 资源下载展示：

同 **news** 类和 **notice** 类，这里定义了一个 **resource** 类，具体如下：

```
private String id,title,publishTime;
4 个用法
private List<Object[]> links;
1 个用法
private static final Map<String, String> RESOURCE_PATHS = new HashMap<>() {{
    put("student", "/static/resource/resource");
    put("teaching", "/static/resource/resource1");
    put("research", "/static/resource/resource2");
    put("basic", "/static/resource/resource3");
}};

public resource() {
}

public resource(String id) {
    this.id = id;
    // 获取资源信息
    List<Object[]> resources = resourceDao.getResourcesById(id);
    if (!resources.isEmpty()) {
        Object[] resource = resources.getFirst();
        this.title = resource[0].toString();
        this.publishTime = resource[1].toString();
    }
    // 获取资源链接
    this.links = resourceDao.getResourcesLink(id);
}

public String getId() { return id; }

public void setId(String id) {
    this.id = id;
    // 获取资源信息
    List<Object[]> resources = resourceDao.getResourcesById(id);
    if (!resources.isEmpty()) {
        Object[] resource = resources.getFirst();
        this.title = resource[0].toString();
        this.publishTime = resource[1].toString();
    }
    // 获取资源链接
    this.links = resourceDao.getResourcesLink(id);
}
```

图 28 resource 类

可以看出和 notice 类实际上是十分相似的，这里不过多解释。前端 JSP 界面也不再展示。

(13)DAO 模块:

数据库连接实际上是对数据库的各个表进行一系列操作，由于不涉及到复杂的查询操作，这里不过多讲述数据库操作。接下来将展示 DAO 模块的基础类 SQLHelper 类。

```
public class SQLHelper {  
    private static final String DRIVER_NAME  
="com.microsoft.sqlserver.jdbc.SQLServerDriver";  
    private final static String URL  
="jdbc:sqlserver://localhost:1433;databaseName=javaEEFinal;trustServerCertificate=  
true";  
  
    private static final String USER = "sa";  
    private static final String PWD = "yyk200412=";  
    private static Connection conn=null;  
    static {  
        try {  
            Class.forName(DRIVER_NAME);  
        } catch (Exception ex) {  
            System.out.println(ex.getMessage());  
        }  
    }  
  
    public static List<Object[]> executeQueryList(String sql){  
        List<Object[]> list= new ArrayList<>();  
        ResultSet rs=null;  
        try {  
            conn = DriverManager.getConnection(URL, USER, PWD);  
            Statement cmd = conn.createStatement();  
            rs=cmd.executeQuery(sql);
```



```

        //元数据
        ResultSetMetaData metaRs= rs.getMetaData();
        int colsNum=metaRs.getColumnCount();
        while(rs.next()){
            Object[] arr=new Object[colsNum];
            for(int i=0;i<arr.length;i++) {
                arr[i]=rs.getObject(i+1);
            }
            list.add(arr);
        }
        conn.close();
    } catch (Exception ex) {
        System.out.println(ex.getMessage());
    }
    return list;
}

public static ResultSet executeQuery(String sql){
    ResultSet rs=null;
    try {
        conn = DriverManager.getConnection(URL, USER, PWD);
        Statement cmd = conn.createStatement();
        rs=cmd.executeQuery(sql);
        conn.close();
    } catch (Exception ex) {
        System.out.println(ex.getMessage());
    }
    return rs;
}

public static int executeUpdateByParams(String sql,Object... params){//可变

```

参数

```
int r=0;

try {
    Connection conn = DriverManager.getConnection(URL, USER,
PWD);

    PreparedStatement cmd=conn.prepareStatement(sql);
    for(int i=0;i<params.length;i++){
        switch (params[i]) {
            case String s -> cmd.setString(i + 1, s);
            case Integer integer -> cmd.setInt(i + 1, integer);
            case Double v -> cmd.setDouble(i + 1, v);
            case null, default -> cmd.setObject(i + 1, params[i]);
        }
    }

    r=cmd.executeUpdate();
    conn.close();
} catch (Exception ex) {
    System.out.println(ex.getMessage());
}

return r;
}

public static int executeUpdate(String sql){
    int r=0;
    try {
        Connection conn = DriverManager.getConnection(URL, USER,
PWD);

        Statement cmd = conn.createStatement();
        r=cmd.executeUpdate(sql);
        conn.close();
    } catch (Exception ex) {
```

```
        System.out.println(ex.getMessage());
    }
    return r;
}
}
```

其中代码的逻辑较为简单，`executeQueryList(String sql)`执行查询 SQL 语句，返回 `List<Object[]>`类型的结果集，适用于需要将查询结果转换为列表形式的场景。`executeQuery(String sql)`执行查询 SQL 语句，直接返回 `ResultSet` 结果集，适用于简单的查询操作。`executeUpdateByParams(String sql, Object... params)`执行带参数的增删改 SQL 语句，其支持动态参数，可以防止 SQL 注入，并返回受影响的行数。`executeUpdate(String sql)`执行不带参数的增删改 SQL 语句，并返回受影响的行数。

5.3 系统展示

静态页面不过多展示。这里只展示几个重要页面。

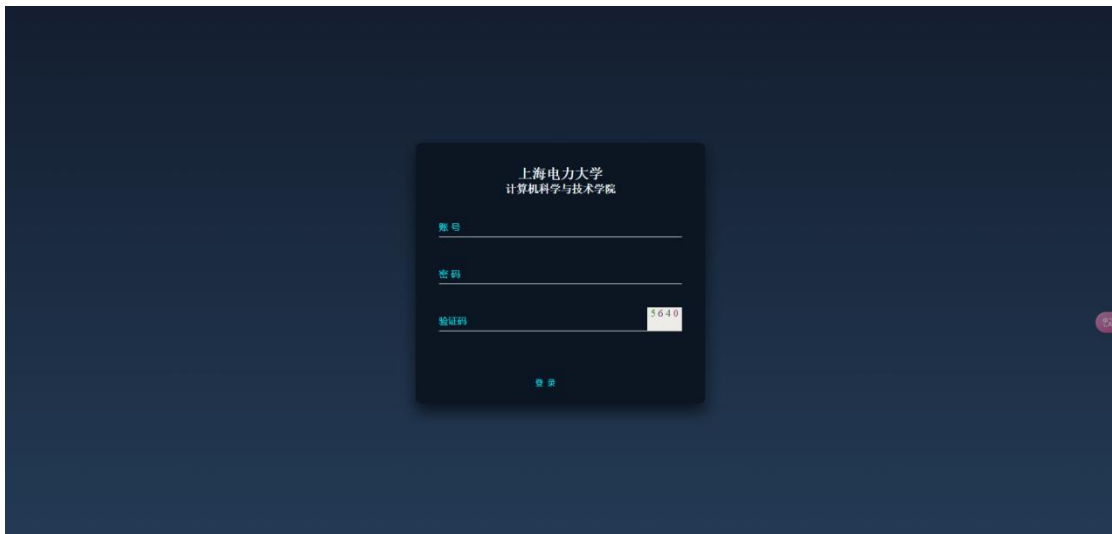


图 29 登录界面

可以看到验证码等的实现。



图 30 首页

首页当中，上面是轮播图，然后是自己实现的学院新闻，可以通过点击按钮实现翻页，然后是仿照学院官网实现的教务通知和通知公告栏。

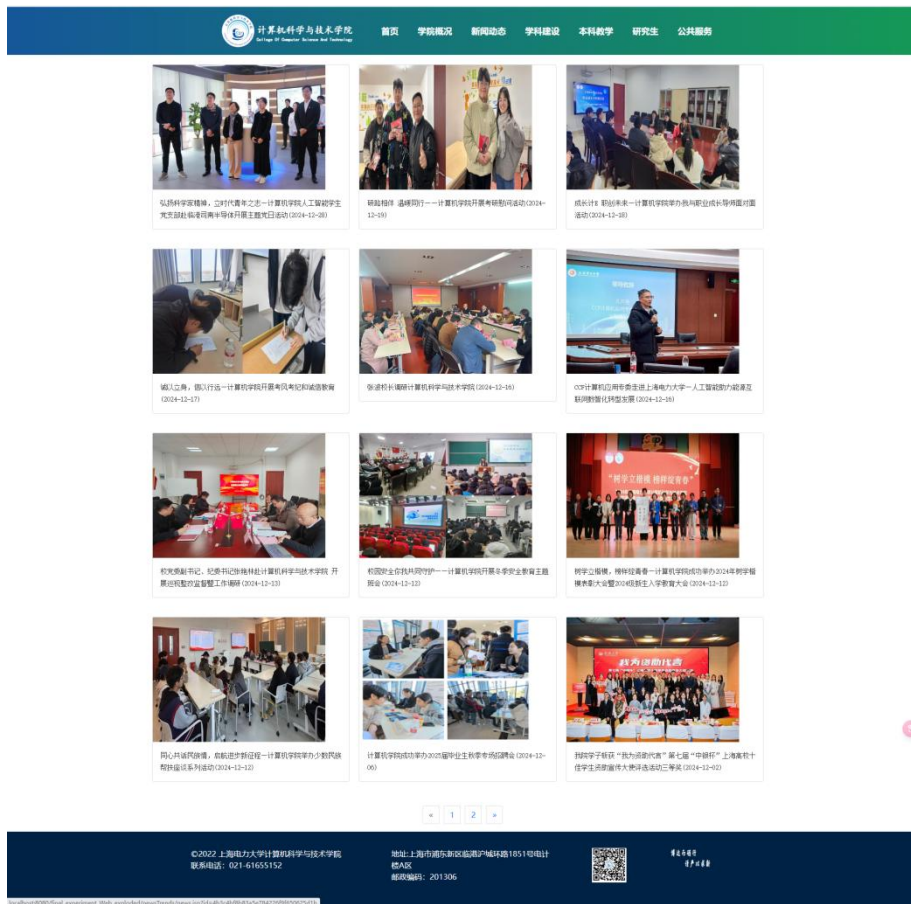


图 31 新闻动态界面

这里展示了新闻。



图 32 具体新闻展示

这里展示了一个具体新闻的界面，下面也可以查看到评论区的一个具体实现。

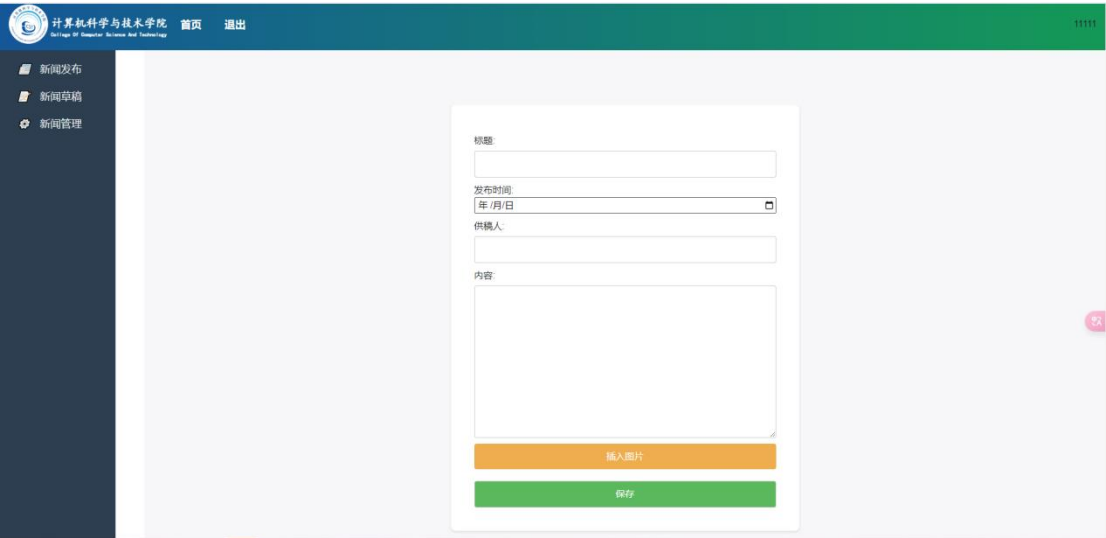


图 33 新闻发布界面

这是新闻管理界面当中的新闻发布。



图 34 新闻管理

这是具体的新闻管理界面。新闻草稿与新闻管理相同，不展示。



图 35 教务通知

这里是教务通知的一个展示界面，通知公告相同，不展示。



图 36 教务通知界面

这是一个有具体资源的教务通知界面，可以实现点击下载教务通知的附件。

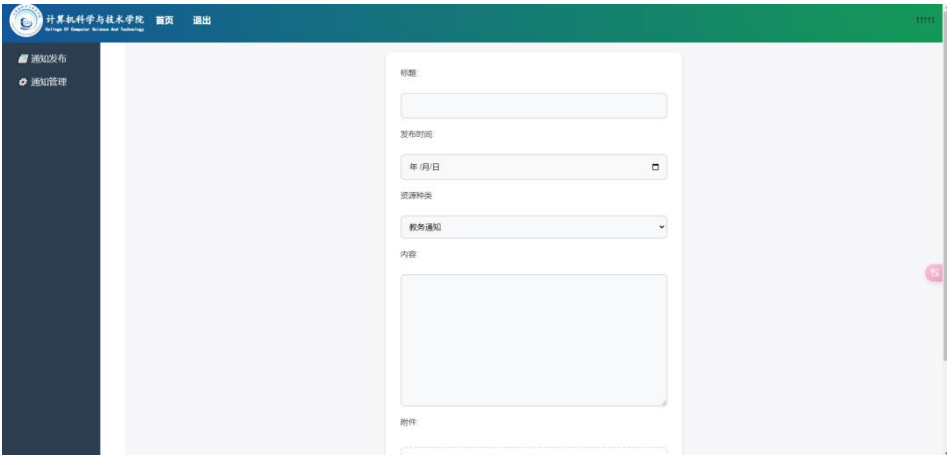


图 37 通知发布

这是通知发布的界面。

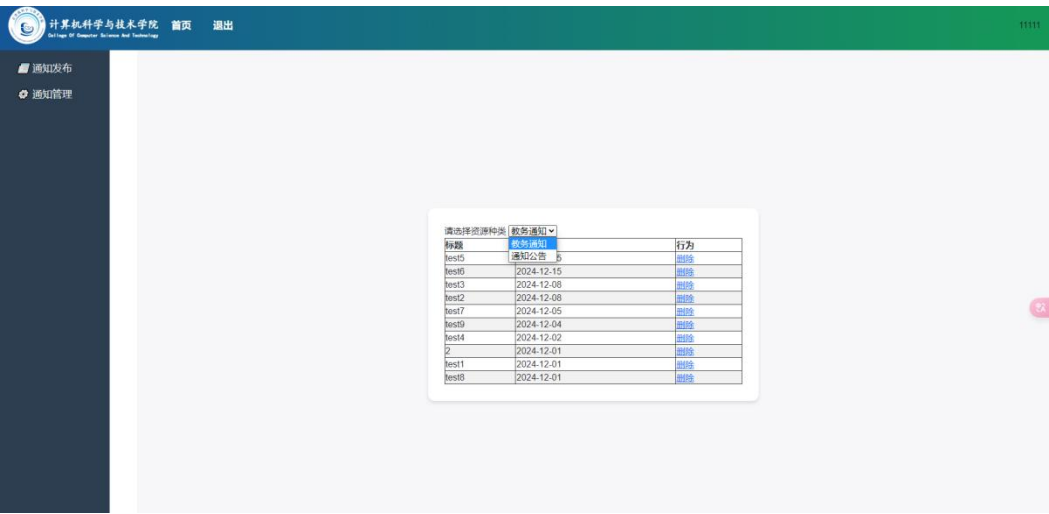


图 38 通知管理

这是通知管理界面。在这里可以实现两个种类的通知的管理。



图 39 资源下载

这是资源下载界面，用户可以选择具体的一类资源进行查看下载。



图 40 资源界面

这是一个具体的资源下载界面，与教务通知极为相似，只是少了正文内容。

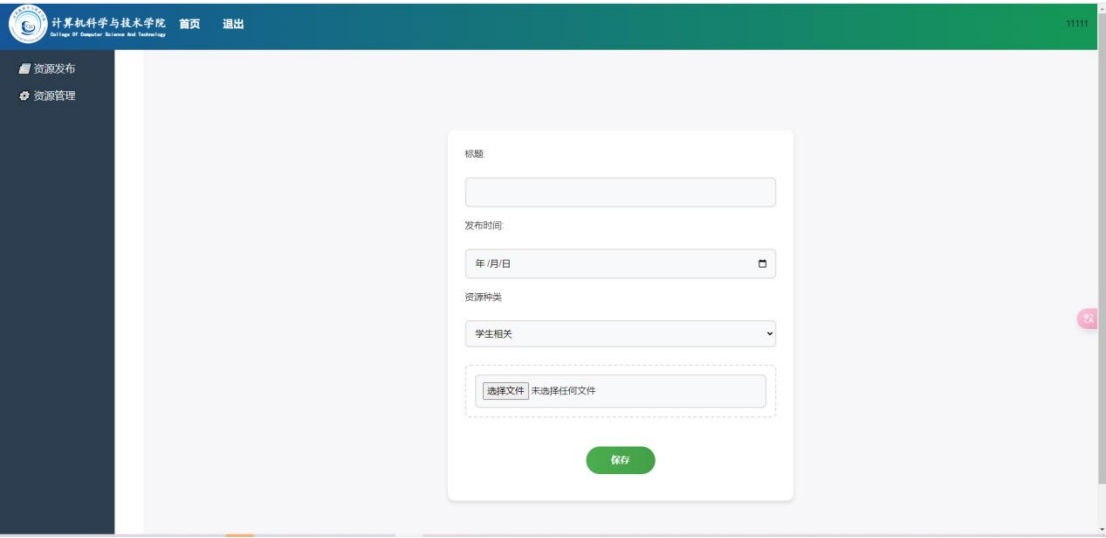


图 41 资源发布界面

这是资源发布界面，管理员可以在这里实现资源的发布。

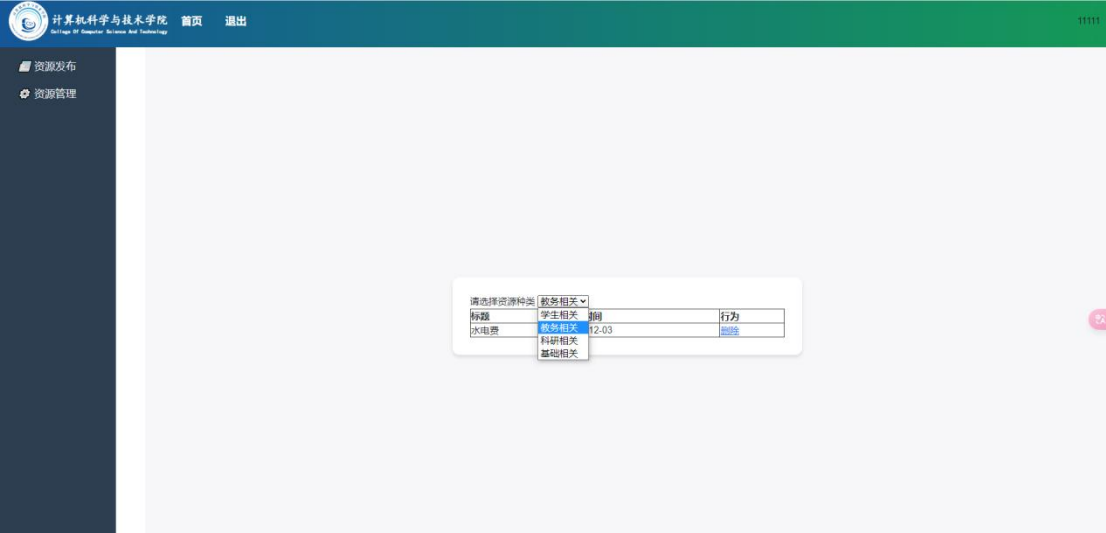


图 42 资源管理

资源管理界面与通知管理界面的 css 是一致的，所以会出现相同的界面。但是从资源种类上可以看出来这是资源管理界面。

5.4 本章小结

本章详细介绍了计算机学院门户网站系统的具体实现过程。首先明确了系统开发环境，包括 IDE、JDK、数据库等必要软件及其版本。随后重点阐述了系统各个关键功能模块的实现细节，包括基础的 MD5 加密和登录过滤、新闻动态的发布管理、评论功能、通知公告、资源下载等模块，并通过 SQLHelper 类实现了统一的数据库操作。在技术实现上，系统采用 JSP+Servlet 架构，结合 JavaScript

实现动态交互，通过 **JavaBean** 简化开发，最终通过系统界面截图直观展示了各个功能模块的实现效果，验证了系统设计的合理性和实用性。

6 总结与展望

6.1 全文总结

本文详细介绍了基于 Java EE 平台的计算机学院门户网站系统的开发过程。项目从需求分析开始，明确了系统的目标用户群体和功能需求，包括游客、普通教职工和学生、管理员等不同角色的功能划分。系统设计阶段，采用了 B/S 架构，明确了表示层、业务逻辑层和数据访问层的三层架构设计，为系统的开发提供了清晰的技术路线图。在数据库设计方面，详细描述了登录表、新闻表、新闻评论表、通知表等数据库表的结构，为系统的数据存储提供了坚实的基础。

系统实现阶段，本文重点介绍了关键功能模块的实现，包括加密模块、登录过滤模块、验证码模块等基础功能，以及新闻动态管理、通知公告管理、资源下载管理等核心业务模块。通过 JavaBean 简化了业务逻辑的处理，通过 JSP+Servlet 架构实现了系统的动态交互，通过 JavaScript 实现了前后端的数据交互。系统测试阶段，通过单元测试和集成测试验证了系统功能的完整性和稳定性。

系统展示部分，通过截图直观展示了系统的主要界面和功能，包括登录界面、首页、新闻动态界面、具体新闻展示界面、新闻发布界面、新闻管理界面、教务通知界面、通知发布界面、通知管理界面、资源下载界面、资源界面、资源发布界面和资源管理界面等，直观地展示了系统的功能实现效果。

6.2 未来展望

本系统虽然完成了计算机学院门户网站的核心功能开发，达到了预期的设计目标，但是改进和提升的空间依然很大，需要继续优化和完善。

从技术层面来讲，本系统是基于 Java EE 平台的单体应用，各模块集成在一起，中间件部分也仅仅使用了数据库连接技术，文件存储部分采用的本地服务器存储。如果访问量增多，性能和存储成本也是一个需要考虑的问题。因此，未来改进或者重构系统时，可以从多方面进行考虑。架构选型方面，可以使用当前流行的微服务架构，将各个模块单独组成微服务，独立部署和运行，能够极大提高可扩展性、可维护性和系统性能。在存储层面，可以自行搭建存储服务，使用单独的存储服务给系统业务提供支持，降低使用本地存储的扩展性问题和潜在成本。在安全层面，使用更先进的加密技术和动态权限控制，来更加灵活地控制系统接口，提高系统的安全性。

从业务层面来讲，本系统完成了从用户登录、新闻发布、通知公告管理、资源下载等一整条业务逻辑线，与数据库管理系统一起，完成这个业务需求。但是跟整个学院的信息化系统相比，如果要投入实用或商业化，还需要更多模块的支持。本系统以新闻动态和资源下载作为出发点，完成信息发布和管理这两个核心的业务功能，并辅以用户管理、访问控制等业务作为支撑。在后期做系统扩展时，可引入在线教育模块、学生互动社区模块、国际版语言支持等。换言之，学院的门户网站是一个综合、复杂且庞大的系统，如果需要完美实现一个学院的门户网站，依然任重道远。