

南京理工大学泰州科技学院

# 毕业设计说明书(论文)

作者： 龙辉 学号： 1909030223

学院(系)： 计算机科学与工程学院

专业： 计算机科学与技术

题目： RSS 阅读器的设计与实现

指导者： 解定东 讲师

评阅者： 丁勇 副教授

2023 年 5 月

# 毕业设计说明书（论文）中文摘要

随着互联网的不断发展，人们越来越依赖于通过网络获取信息。RSS 阅读器作为一种重要的 RSS 客户端，可以方便地订阅、阅读和管理多个 RSS 源，成为人们获取所需信息的有力工具。本文旨在设计和实现一个基于 Web 的 RSS 阅读器，满足用户多样化的订阅和阅读需求。首先，本文对 RSS 技术和阅读器功能进行了详细介绍和分析，然后根据用户需求设计了阅读器的功能和界面。接着，采用了常用的 Web 开发技术和框架，实现了一个完整的 RSS 阅读器，并对其进行了详细的测试和优化。最后，对阅读器进行了功能和性能评估，证明了其在用户订阅、阅读、管理等方面的优越性和可靠性。本文的贡献在于提供了一种可靠、高效、易用的 RSS 阅读器，为用户获取所需信息提供了方便和效率。同时，本文的设计和实现过程也为其他开发者提供了参考和借鉴，推动了 RSS 阅读器的发展和应用。

**关键词** RSS 阅读器 订阅 信息获取

# 毕业设计说明书（论文）外文摘要

**Title**      Design and Implementation of the RSS Reader

---

## **Abstract**

With the continuous development of the Internet, people are increasingly relying on obtaining information through the network. As an important type of RSS client, RSS reader can conveniently subscribe, read, and manage multiple RSS sources, becoming a powerful tool for people to obtain necessary information. This thesis aims to design and implement a web-based RSS reader to meet users' diversified subscription and reading needs. Firstly, this thesis provides a detailed introduction and analysis of RSS technology and reader functions, and then designs the reader's functions and interface based on user needs. Next, the commonly used web development technologies and frameworks are adopted to implement a complete RSS reader, which is thoroughly tested and optimized. Finally, the reader is functionally and performance-evaluated, proving its superiority and reliability in user subscription, reading, and management. The contribution of this thesis is to provide a reliable, efficient, and user-friendly RSS reader, which provides convenience and efficiency for users to obtain necessary information. At the same time, the design and implementation process of this thesis also provides reference and inspiration for other developers, promoting the development and application of RSS readers.

**Keywords**   RSS   Reader   Subscription   Information Retrieval

## 目 录

|                         |    |
|-------------------------|----|
| 1 绪论 .....              | 1  |
| 1.1 选题背景分析 .....        | 1  |
| 1.2 国内外发展现状 .....       | 2  |
| 1.3 本文的主要工作 .....       | 2  |
| 1.4 本文的组织结构 .....       | 2  |
| 2 相关技术简介 .....          | 4  |
| 2.1 JavaEE 技术 .....     | 4  |
| 2.2 Spring MVC 框架 ..... | 4  |
| 2.3 eclipse IDE .....   | 4  |
| 2.4 Oracle 数据库 .....    | 5  |
| 2.5 ROME 框架 .....       | 5  |
| 3 需求分析 .....            | 6  |
| 3.1 可行性分析 .....         | 6  |
| 3.2 系统功能需求 .....        | 6  |
| 4 软件系统设计 .....          | 8  |
| 4.1 系统总体设计 .....        | 8  |
| 4.2 数据库设计 .....         | 9  |
| 4.3 详细设计 .....          | 13 |
| 4.4 系统流程设计 .....        | 14 |
| 4.5 系统类设计 .....         | 16 |
| 4.6 系统活动图设计 .....       | 20 |
| 5 系统实现 .....            | 23 |
| 5.1 工具模块 .....          | 23 |
| 5.2 用户使用功能模块 .....      | 24 |
| 6 系统测试 .....            | 43 |
| 6.1 用户登录功能测试 .....      | 43 |
| 6.2 用户注册功能测试 .....      | 43 |
| 6.3 主页分组和文章更新测试 .....   | 43 |
| 6.4 RSS 源管理功能测试 .....   | 44 |
| 6.5 文章列表和阅读功能测试 .....   | 44 |
| 6.6 收藏历史功能测试 .....      | 45 |
| 6.7 历史阅读功能测试 .....      | 45 |
| 结束语 .....               | 46 |
| 致 谢 .....               | 47 |
| 参 考 文 献 .....           | 48 |

## 1 绪论

随着互联网的不断发展，人们越来越依赖于通过网络获取信息。网络上的许多网站提供了各种各样的信息和新闻，但是如果我们想要及时地获取这些信息，每天都要花费大量的时间去浏览这些网站。为了解决这一问题，RSS 技术应运而生。RSS 阅读器作为一种重要的 RSS 客户端，可以方便地订阅、阅读和管理多个 RSS 源，成为人们获取所需信息的有力工具。

RSS(Really Simple Syndication)是一种用于发布和订阅信息的格式，通过使用 RSS，用户可以在不访问网站的情况下及时获取网站上的更新内容<sup>[1]</sup>。这种技术不仅可以提高用户的信息获取效率，还可以减少网络流量，降低网站的访问压力。

### 1.1 选题背景分析

RSS 阅读器是一种能够阅读 RSS 和 Atom 格式文档的软件。RSS（简易信息聚合，也叫 Really Simple Syndication、聚合 RSS、聚合内容），是一种消息来源格式规范，用以聚合经常发布更新数据的网站，例如博客文章、新闻、音频或视频的网摘，它具有时效性强，生产成本低廉，定位准确的特点<sup>[2]</sup>。RSS 文件是一个基于 XML 的文件<sup>[3]</sup>，通常被用于新闻和其他按顺序排列的网站<sup>[4]</sup>。它创建一个订阅源，该订阅源定期将新闻文章和其他订阅信息从网站传输到它的用户。

RSS 阅读器的历史可以追溯到 2000 年左右，随着社交媒体和移动设备的兴起，RSS 阅读器的地位逐渐下降。在中国，这种高效的阅读方式并未普及。除了一些博客和主流媒体网站外，大多数人仍然不熟悉 RSS。RSS 的目标受众是具有定向阅读习惯的人，但 RSS 源的质量一直好坏参半。门户网站的信息太多，小网站信息太少，使得 RSS 用户难以增长<sup>[1]</sup>。目前流行的信息主要是 BLOG 和分类新闻。国内 RSS 内容提供商并不多。国内较为受欢迎的 RSS 阅读器包括摸鱼 kik 和 Ego Reader。

总之，虽然 RSS 阅读器的地位已经不再像过去那样重要，但它仍然是一种方便实用的工具，能够帮助用户更高效地获取和管理信息。特别是对于需要关注多个网站的用户或组织，使用 RSS 阅读器可以大大提高工作效率和信息获取质量<sup>[5]</sup>。同时，随着技术的不断发展，RSS 阅读器也在不断更新和演进，例如一些阅读器已经开始支持社交媒体集成、智能推荐等功能，这些新功能让 RSS 阅读器的价值更加显著。

## 1.2 国内外发展现状

### 1.2.1 国外发展现状

早期, RSS 主要被用于博客的订阅。随着互联网技术的发展, RSS 技术和阅读器已经成为了一种重要的信息获取和管理工具, 被广泛应用于新闻、博客、社交媒体等领域。随着技术的不断发展和变革, RSS 技术和阅读器也在不断地演进和完善, 为用户提供更加便捷、高效、个性化的服务。目前, 许多大型的互联网公司, 如 Google、Apple 等, 都会提供自己的 RSS 服务。Google Reader 曾是最为知名的 RSS 订阅器之一, 但在 2013 年关闭。现在, Inoreader、Feedly 等 RSS 订阅器已经成为了主流<sup>[5]</sup>。

### 1.2.2 国内发展现状

国内的 RSS 技术起步较晚, 但近年来也得到了快速发展, RSS 技术被越来越多地应用于其他领域, 如企业档案管理<sup>[6]</sup>、数学信息共享平台<sup>[7]</sup>、公安系统<sup>[8]</sup>等。目前, 国内主流的 RSS 订阅器有摸鱼 kik 和 Ego Reader 等。国内的一些互联网公司也推出了自己的 RSS 订阅器产品, 摸鱼 kik 为搜狐公司推出。在中国, RSS 技术虽然不如国外发展得那么成熟, 但是随着越来越多的用户开始关注信息的订阅和推送, RSS 技术的应用前景仍然非常广阔。

总的来说, 随着移动设备的普及和互联网技术的不断发展, RSS 技术的应用范围将会越来越广泛。未来, RSS 技术将继续在信息订阅和推送领域发挥重要作用<sup>[9]</sup>。

## 1.3 本文的主要工作

本文的主要工作是设计和实现一个基于 JavaEE 技术和 Spring MVC 框架的 RSS 阅读器, 旨在提供一个高效、可靠、易用的 RSS 订阅和阅读工具。为了实现这一目标, 本文按照学术论文的要求, 对相关技术和工具进行了介绍, 并详细阐述了需求分析、软件系统设计、系统实现和系统测试等环节。

## 1.4 本文的组织结构

绪论部分主要介绍了论文的选题背景、研究意义和国内外发展现状, 并对本篇论文的研究内容进行了概述。

相关技术简介部分介绍了本篇论文所采用的 JavaEE 技术、Spring MVC 框架、Eclipse IDE、Oracle 数据库和 ROME 框架等技术及工具。

需求分析部分首先进行了可行性研究, 包括经济可行性和技术可行性, 并对系统功能需求进行了详细阐述。

软件系统设计部分包括系统总体设计、数据库设计、详细设计、系统流程设计、系统类设计和系统活动图设计等内容。

系统实现部分主要介绍了工具模块和用户使用功能模块，并详细描述了用户登录、注册、退出登录、功能主界面、RSS 源管理、文章列表、文章收藏历史和文章阅读历史等功能。

系统测试部分通过对用户登录功能、主页分组和文章更新、RSS 源管理、文章列表和阅读、收藏历史和历史阅读等功能进行测试和验证，保证了系统的稳定性和可靠性。

结束语部分对完成毕业设计论文的工作进行了总结。

## 2 相关技术简介

### 2.1 JavaEE 技术

JavaEE (Java Enterprise Edition) 是一种基于 Java 语言的企业级应用程序开发平台。它提供了一系列的 API 和规范,使得开发者可以快速、高效地开发和部署集成、安全、可靠的企业级应用程序<sup>[10]</sup>。JavaEE 技术包括各种组件和技术,例如 Servlet、JSP、EJB、JPA、JMS、Web Services 等,这些组件和技术可以用于构建分布式、多层次、面向服务的应用程序。JavaEE 平台还提供了各种应用服务器和开发工具,例如 Tomcat、WebLogic、WebSphere、Eclipse 等,这些工具可以帮助开发者更加方便地开发和部署 JavaEE 应用程序。JavaEE 技术的优点包括可移植性、安全性、可扩展性、可靠性、高效性等,使得它成为开发企业级应用程序的首选技术之一。

### 2.2 Spring MVC 框架

Spring MVC 是基于 Java 的开源 Web 框架,它是 Spring Framework 的一部分,用于构建 Web 应用程序。Spring MVC 框架采用了 MVC (Model-View-Controller) 模式,将应用程序分为模型、视图和控制器三个部分,实现了业务逻辑和用户界面的分离。模型 (Model) 用于封装应用程序的数据,视图 (View) 用于呈现数据,控制器 (Controller) 用于处理用户请求和响应<sup>[11]</sup>。Spring MVC 框架的核心组件包括 DispatcherServlet、HandlerMapping、Controller、ViewResolver 等,其中 DispatcherServlet 是整个框架的核心部分,用于接收并分发 HTTP 请求,HandlerMapping 用于将请求映射到相应的控制器,Controller 负责处理业务逻辑并返回响应,ViewResolver 用于将模型数据呈现到视图上<sup>[12]</sup>。Spring MVC 框架的优点包括易用性、灵活性、可扩展性和可测试性等,它可以与其他 Spring 框架和第三方库集成,提供了很多功能和特性,使得开发 Web 应用程序更加容易和高效。

### 2.3 eclipse IDE

在开发我的 RSS 阅读器时,考虑到系统需求,我决定使用 Eclipse 作为开发工具。Eclipse 是一个开源的集成开发环境,专门用于 Java 开发,提供了框架和服务,可以通过插件组件构建定制的开发环境。Eclipse 默认带有标准的插件,比如 Java 开发工具 (JDT),这些插件可以帮助开发人员快速地开发 Java 应用程序。此

外，Eclipse 还支持安装各种插件，包括语言、框架和工具的支持，这些插件可以帮助开发人员构建企业级集成开发环境，并且更高效地完成开发任务。

## 2.4 Oracle 数据库

Oracle 数据库是一款商业关系数据库管理系统，由甲骨文公司(Oracle)开发。它以其高性能、高可靠性和强大的功能而闻名，成为最流行的大型数据库之一。Oracle 数据库被广泛应用于各类大型网站和企业应用程序中，它可以运行在多种平台上，支持面向对象和丰富的数据类型，并且在各种操作系统平台的操作基本一致<sup>[13]</sup>。此外，Oracle 还提供了 SQL Developer 作为管理和开发工具，其图形界面不仅降低了技术要求门槛，而且极大地提高了开发速度。

## 2.5 ROME 框架

ROME 是一个用于 RSS 和 Atom 源的 Java 框架，它是开源的，根据 Apache 2.0 许可证授权<sup>[14]</sup>。ROME 包括一组用于各种联合馈送的解析器和生成器，以及用于从一种格式转换为另一种格式的转换器。解析器可以返回 Java 对象，这些对象既可以针对您要使用的格式，也可以是通用的标准化 SyndFeed 类，可以在不考虑传入或传出 feed 类型的情况下处理数据。

ROME 还包括几个子项目，包括 Modules、Fetcher（已弃用）、Propono 和 OPML。Modules 提供对馈送扩展（如 GeoRSS、iTunes、Microsoft SSE 和 SLE、Google GData 等）的支持。Fetcher 是一个缓存馈送获取器，支持通过 HTTP 条件 GET 检索馈送。Propono 支持发布协议，特别是 Atom 发布协议和旧版 MetaWeblog API。Propono 包括一个 Atom 客户端库、一个 Atom 服务器框架和一个支持 Atom 协议和 MetaWeblog API 的博客客户端。OPML 是 Outline Processor Markup Language (OPML) 解析器和工具。

## 3 需求分析

### 3.1 可行性分析

#### 3.1.1 经济可行性

从经济角度来看，设计和实现一个基于 Web 的 RSS 阅读器具有良好的经济可行性。首先，相较于传统的纸质媒体或者其他电子媒体，RSS 阅读器具有低成本、高效率的特点。其次，由于 Web 技术的普及和成熟，开发和维护一个基于 Web 的 RSS 阅读器的成本也相对较低。另外，RSS 阅读器也可以通过广告等方式进行商业化运营，具有较好的盈利潜力。

#### 3.1.2 技术可行性

从技术角度来看，设计和实现一个基于 Web 的 RSS 阅读器具有良好的技术可行性。首先，RSS 技术已经成熟，实现一个基于 RSS 的阅读器相对简单。其次，Java 的 Rome 库提供了丰富的 RSS 处理工具，可以方便地实现 RSS 数据的解析和处理。再次，使用 SpringMVC 作为后端框架可以方便地实现 MVC 架构，提高代码的可维护性和可扩展性。同时，使用 Oracle 数据库可以提供稳定、高效的数据存储和查询能力，确保阅读器的数据可靠性和性能。最后，Bootstrap 作为前端框架可以帮助开发者实现美观、响应式的用户界面，提高用户的体验和使用效率。综合来看，这些技术的应用使得基于 Web 的 RSS 阅读器的设计和实现更加方便、高效、稳定。

#### 3.1.3 操作可行性

从用户操作角度来看，设计和实现一个基于 Web 的 RSS 阅读器也具有良好的可行性。首先，基于 Web 的 RSS 阅读器具有跨平台、无需安装等优势，用户可以在不同设备和系统上使用。其次，通过合理的界面设计和用户体验优化，可以使得用户的订阅、阅读和管理过程更加简单、高效、直观。最后，阅读器的数据同步和自动更新功能可以减少用户的操作量，提高用户的使用效率和体验。

### 3.2 系统功能需求

对于 RSS 阅读器用户的需求进行分析之后，RSS 阅读器应具有以下功能：

(1)RSS 源的管理和订阅功能：允许用户添加、删除、编辑和订阅 RSS 源，为 RSS 源分组，并可以自动更新订阅的 RSS 源。

(2)RSS 文章的浏览和管理功能：提供一个清晰、易用的界面，允许用户查看、

管理 RSS 文章，包括收藏、跳转原网页打开、标记为已读或未读。

(3) 用户账号的管理功能：允许用户注册、登录、注销和修改个人信息，同时可以保护用户的密码安全。

(4) 跨平台和移动设备支持功能：允许用户在不同的平台和移动设备上使用阅读器，并可以同步数据和设置，以提高用户的使用效率和便捷性。

以下为 RSS 阅读器的用例图：

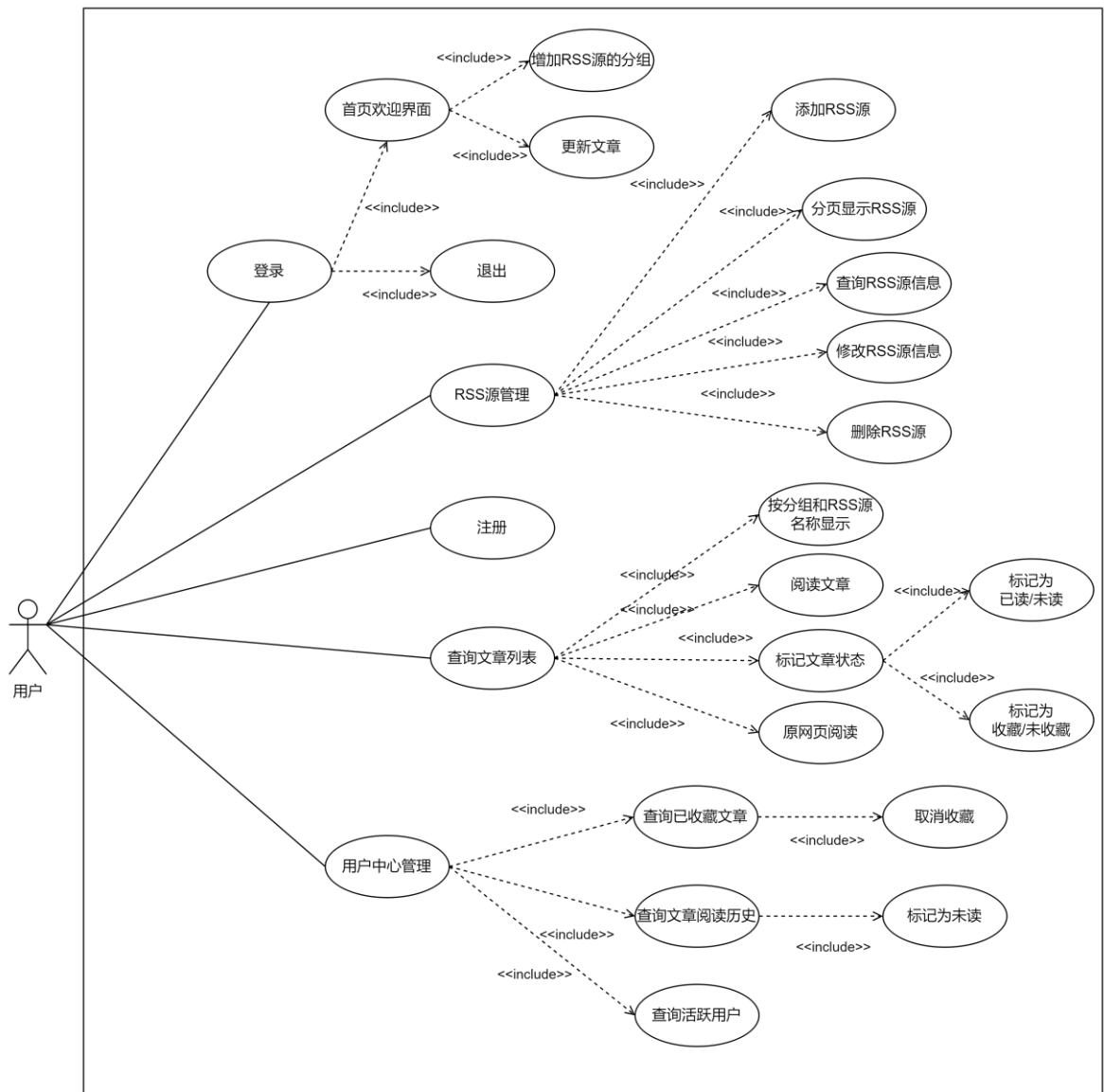


图 3.1 RSS 阅读器用户用例图

## 4 软件系统设计

### 4.1 系统总体设计

#### 4.1.1 用户功能模块

RSS 阅读器用户共有九个功能模块。用户可以在访问客户端网页后进行注册和登录，登录成功后跳转到主界面，可以为 RSS 源添加分组和点击更新文章，点击侧边栏的全部 RSS 对 RSS 源进行增加、删除、修改、查询的管理，用户也可以点击各种分组中的 RSS 的名称查询文章列表来显示相应的文章，能够当前页阅读或者跳转文章的原网页阅读，可以标记文章的状态为已读或未读、已收藏或未收藏，用户可以查询文章的收藏列表和历史阅读列表，功能和查询文章列表相似。点击活跃用户可以显示登录次数较多的用户。功能模块如图 4.1 所示。

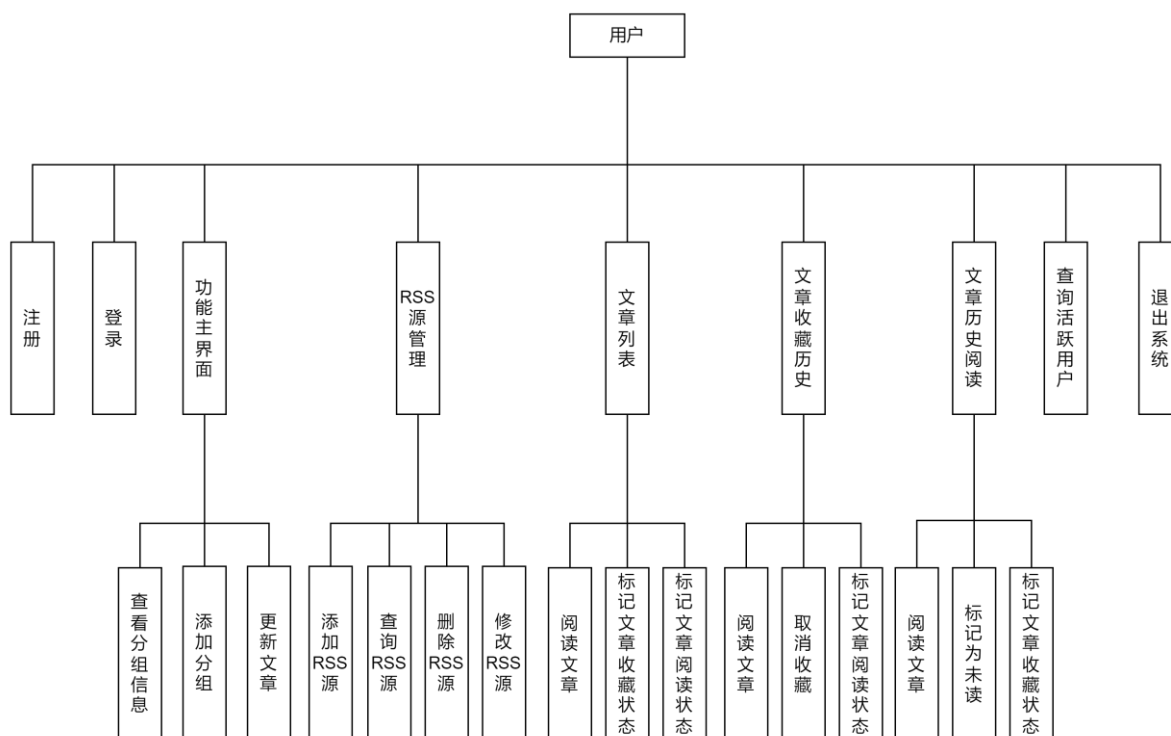


图 4.1 用户功能模块图

#### 4.1.2 系统部署图设计

在设计与实现 RSS 阅读器时，一个重要的方面是如何将其部署在实际的服务器环境中。为此，需要考虑多个因素，如服务器的硬件配置、操作系统和 Web 服务器等。此外，还需要考虑如何将应用程序部署在服务器上，并确保其能够高效地运行和提供服务。为了更好地理解 RSS 阅读器的部署过程，可以通过绘制一个部署图来

描述它的整个部署过程。如图 4.2 所示，在这张部署图中，可以看到各个组件之间的关系，以及它在服务器上的部署方式。这张部署图对于理解和管理 RSS 阅读器的部署过程很有帮助，确保应用程序能够在生产环境中稳定运行。

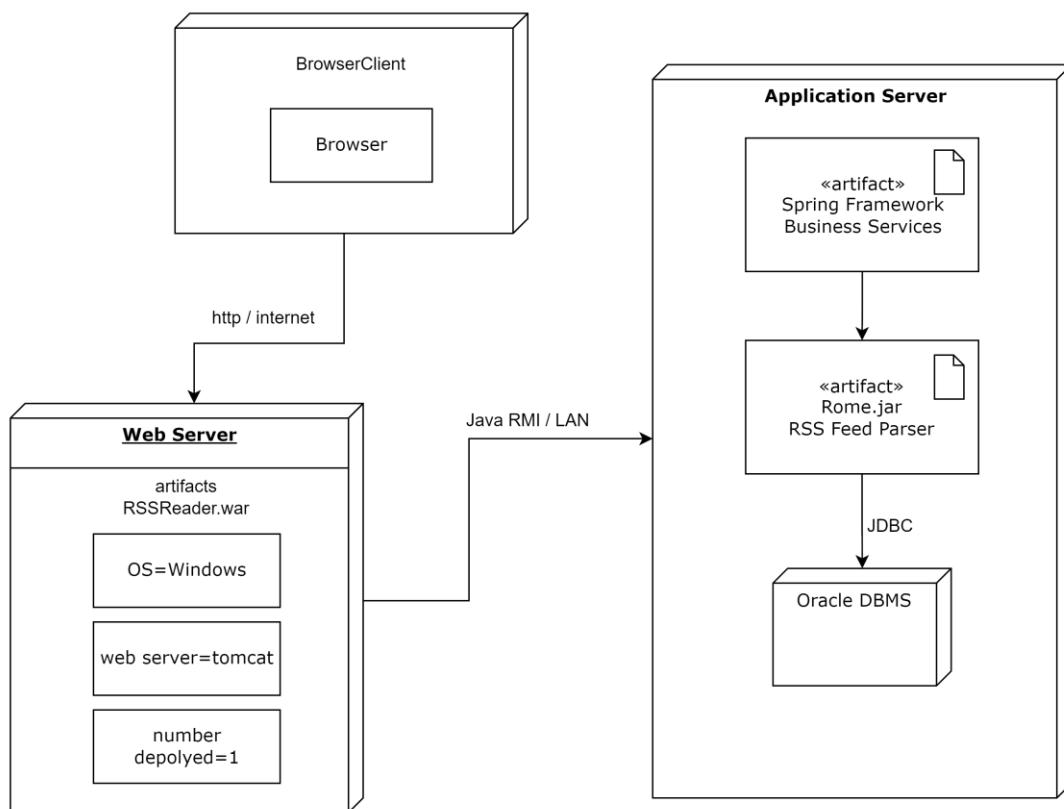


图 4.2 系统部署图

## 4.2 数据库设计

### 4.2.1 概念结构设计

概念设计是为满足数据库系统的需求而进行的一种数据分析和规划过程，其主要从用户的角度出发，对需要存储的数据进行抽象概念建模。通过这种方式，可以建立一个数据模型，以提供基础数据结构和逻辑规则，以便在后续的开发过程中使用。该模型以实体-关系（E-R）图为主要表达方式，用于描述数据的结构、属性、关系及其约束等信息。E-R 图能够清晰地反映实体属性之间的关联关系，包括一对一、一对多、多对多等不同的关系类型。本系统实体 E-R 图如图 4.3 所示。

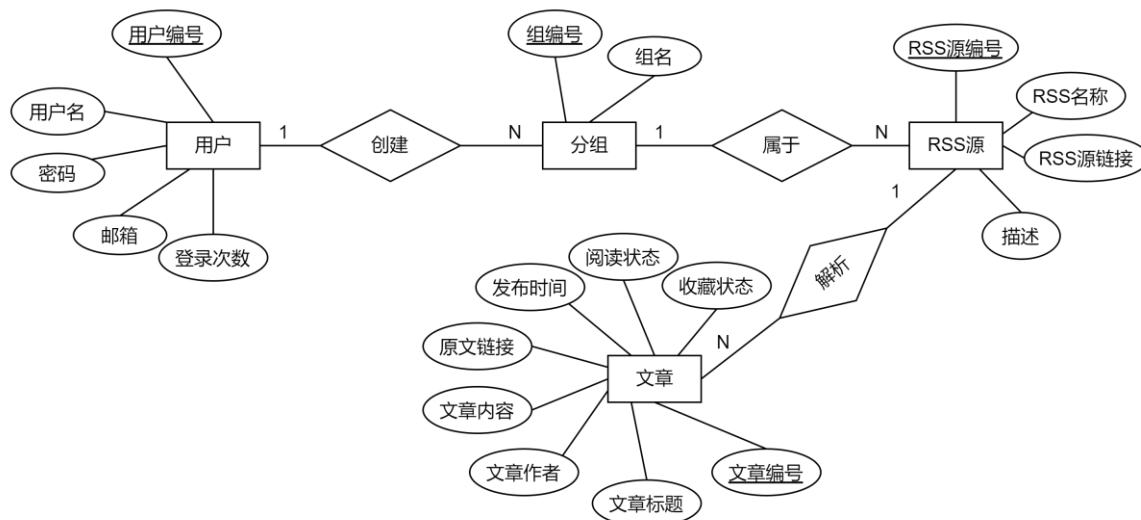


图 4.3 系统实体 E-R 图

#### 4.2.2 逻辑结构设计

在数据库设计的过程中，逻辑结构设计是关键的一步。其中，基本表是一种常见的设计方法，用于描述系统中与数据相关的逻辑结构。该方法主要基于数据的特征，将概念模型转换成关系模型，以实现数据的有效管理和使用。

在基本表的设计中，需要将概念模型转换成实际的数据表，确定表中的各个字段和键，使得数据表能够清晰地表达数据之间的关系和约束。具体而言，需要根据业务需求和数据特征，将数据分解为各个基本表，并进行归纳和分类，以便于数据的规范和管理。

表 4.1 至 4.4 展示了基于基本表设计的数据模型，其中每个数据表都包含了各个字段以及默认值、约束等信息，以便于数据的管理和查询。同时，这些基本表还通过各种键和关联关系相互连接，形成了整个数据模型，在数据的存储和使用过程中发挥着重要的作用。

(1) 用户实体关系模式如下。

users (id、username、password、email、login\_count)

(2) 分组实体关系模式如下。

groups (id、name、user\_id、created\_at)

(3) RSS 源实体关系模式如下。

subscribe (id、url、name、description、user\_id、group\_id)

(4) 文章实体关系模式如下。

article (id、user\_id、sub\_id、title、content、pubdate、link、author、

read、favorite、ubdateat)

表 4.1 用户信息表(users)

| 字段名         | 数据类型          | 是否为空 | 含义   | 备注    |
|-------------|---------------|------|------|-------|
| id          | number(38,0)  | 否    | 用户编号 | 主键    |
| username    | varchar2(100) | 否    | 用户名  |       |
| password    | varchar2(100) | 否    | 用户密码 |       |
| email       | varchar2(100) | 否    | 邮箱   |       |
| login_count | number(38,0)  | 是    | 登录计数 | 默认值 1 |

表 4.2 RSS 源分组信息表(groups)

| 字段名        | 数据类型         | 是否为空 | 含义   | 备注 |
|------------|--------------|------|------|----|
| id         | number(38,0) | 否    | 分组编号 | 主键 |
| name       | varchar2(50) | 否    | 组名   |    |
| user_id    | number(38,0) | 否    | 用户编号 |    |
| created_at | timestamp(6) | 是    | 创建时间 |    |

表 4.3 RSS 源订阅信息表(subscribe)

| 字段名         | 数据类型           | 是否为空 | 含义      | 备注 |
|-------------|----------------|------|---------|----|
| id          | number(38,0)   | 否    | 订阅编号    | 主键 |
| url         | varchar2(1024) | 否    | RSS 源链接 |    |
| name        | varchar2(255)  | 否    | RSS 名称  |    |
| description | varchar2(1024) | 是    | 描述      |    |
| user_id     | number(38,0)   | 是    | 用户编号    |    |
| group_id    | number(38,0)   | 是    | 组编号     |    |

表 4.4 文章信息表(article)

| 字段名      | 数据类型           | 是否为空 | 含义   | 备注    |
|----------|----------------|------|------|-------|
| id       | number(38,0)   | 否    | 文章编号 | 主键    |
| user_id  | number(38,0)   | 否    | 用户编号 |       |
| sub_id   | number(38,0)   | 否    | 订阅编号 |       |
| title    | varchar2(1024) | 否    | 文章标题 |       |
| content  | clob           | 否    | 文章内容 |       |
| pubdate  | timestamp(6)   | 否    | 发布时间 |       |
| link     | varchar2(1024) | 否    | 文章链接 |       |
| author   | varchar2(255)  | 是    | 文章作者 |       |
| read     | number(1,0)    | 是    | 是否阅读 | 默认值 0 |
| favorite | number(1,0)    | 是    | 是否收藏 | 默认值 0 |
| ubdateat | timestamp(6)   | 是    | 更新时间 |       |

4. 2. 3 物理结构设计

物理结构设计是为了实现数据库系统的物理存储和访问而进行的一种设计。在这个过程中，需要考虑数据文件、索引文件、日志文件等的存储位置和存取方式，以便实现高效的数据访问和管理。本系统中,使用 Oracle 数据库来管理数据。Oracle 数据库是一种关系型数据库管理系统，它提供了强大的数据管理功能，能够满足大型企业级应用的需求。为了方便管理数据，使用 SQL Developer 工具来连接和操作 Oracle 数据库。

在本系统中，数据文件存储在 E:\app\LongHui\oradata\orcl 目录下，所属表空间为 USERS，日志文件存储在 E:\app\LongHui\oradata\orcl 目录下。数据文件是数据库中存储实际数据的文件，它包含了表、索引、序列等对象的数据。日志文件则用于记录数据库中所有更改操作的详细信息，在系统开发过程中经常备份数据文件和日志文件到云盘，以便在发生故障时能够快速恢复数据。

物理结构设计对于数据库系统的安全性、完整性、性能和可靠性都有着至关重要的影响。通过这种设计，可以保证数据的安全性和完整性，并提高数据库的性能和可靠性。

### 4.3 详细设计

为了更好地描述系统中各个对象之间的交互过程，本文采用了序列图来进行建模。序列图是一种用于建模对象之间交互的 UML 图，它描述了对对象之间发送消息的时间顺序，以显示多个对象之间的动态协作。如图 4.4 所示，该序列图描述了用户在 RSS 源管理过程中添加 RSS 源时，会查询 RSS 源数据，然后点击添加按钮后界面显示 RSS 源的各项信息的输入框和选项，在输入框中输入相应信息和选择对应选项后点击提交，成功后回到 RSS 源管理页面即可查看添加的信息。如图 4.5 所示，该序列图描述了用户在阅读文章前进行更新文章操作时，会先在数据库中查询该用户添加的 RSS 源信息中的 RSS 链接，然后调用 ROME 库来解析 RSS，处理解析结果后与数据库中原有的文章进行比较去重，将新的文章存储到数据库中，页面上显示更新的文章数量。

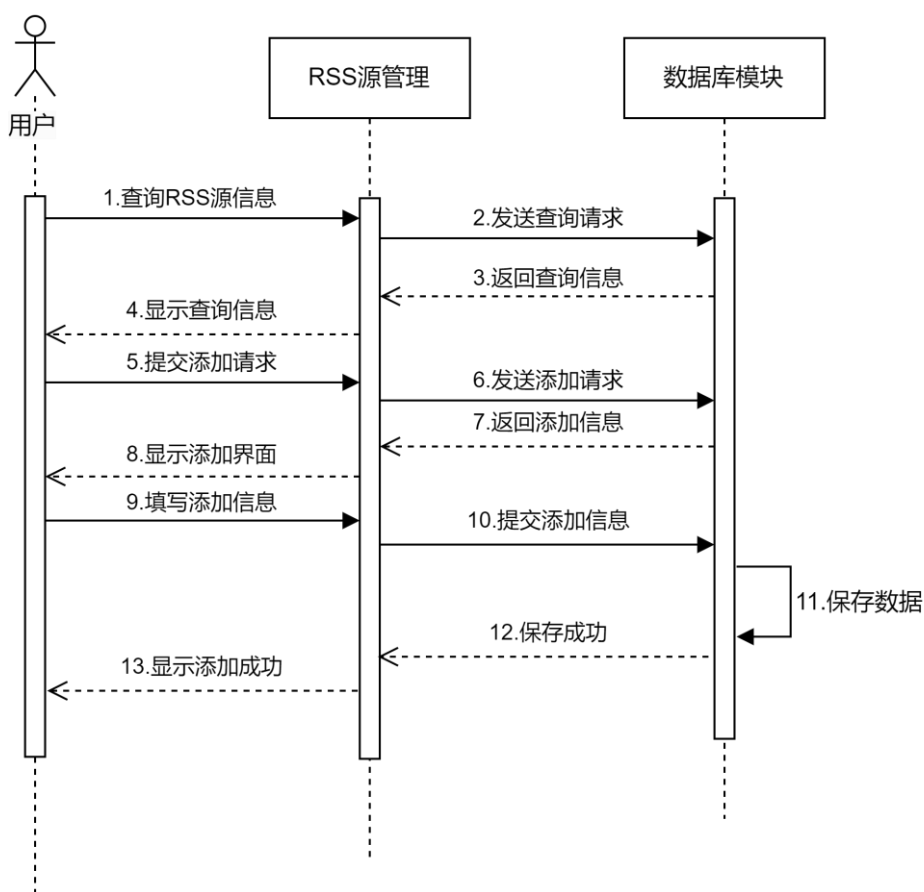


图 4.4 用户添加 RSS 源信息序列图

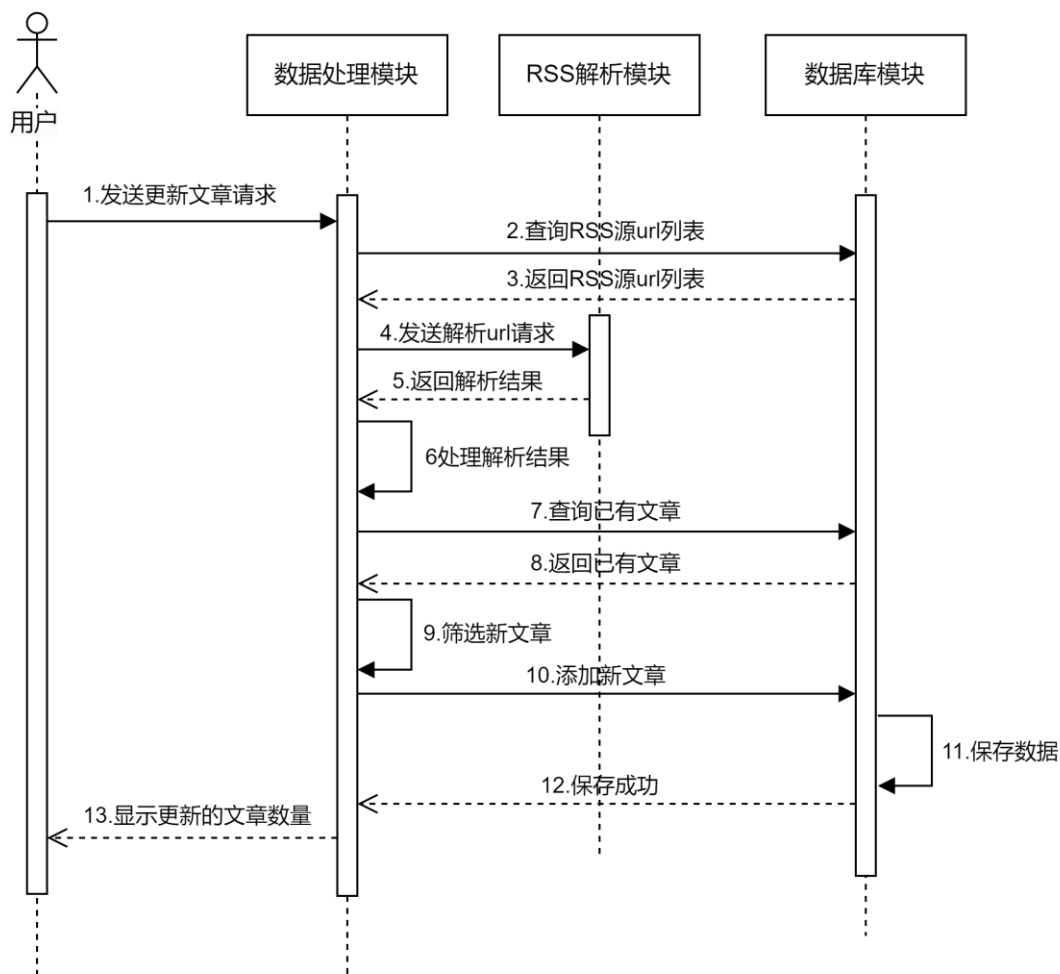


图 4.5 用户更新文章信息序列图

#### 4.4 系统流程设计

RSS 阅读器的使用者可以在客户端网页上登录或注册。成功登录后，即可进入 RSS 阅读器的主界面，开始使用其主要功能。可以在侧边栏查看分组信息，在主页添加新的分组或更新文章，管理全部的 RSS 源，查询文章列表，查询收藏历史和阅读历史，查询 RSS 阅读器活跃用户等。

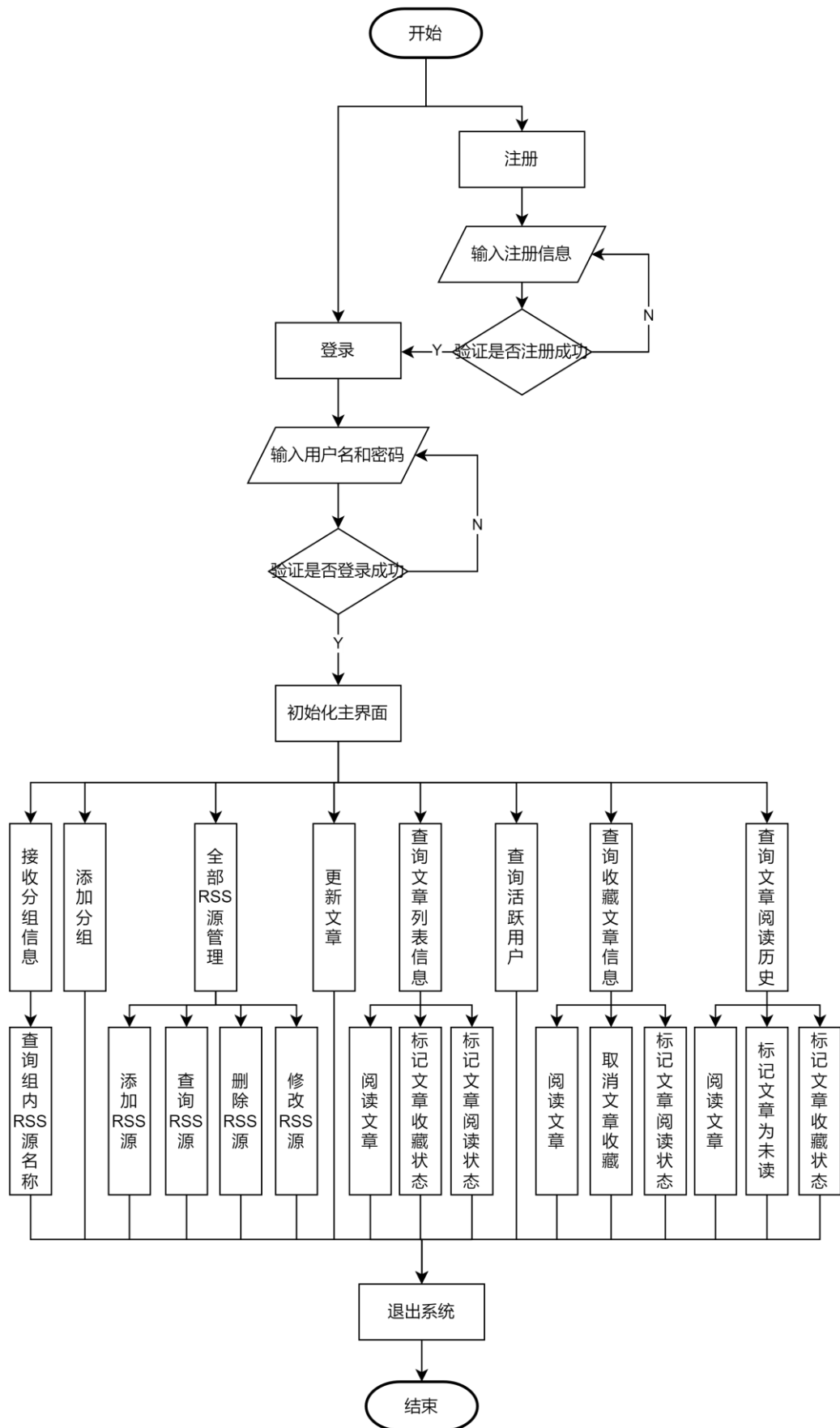


图 4.6 用户功能流程图

## 4.5 系统类设计

### 4.5.1 实体类类图设计

#### (1)Users 类

它包含了私有字段来存储用户的 ID、用户名、密码和电子邮件，以及相应的 getter 和 setter 方法来访问和修改这些字段。

除了这些基本的数据管理方法之外，“Users”类还包含了几种在数据库中查询、删除和更新用户数据的方法。这些方法包括：

queryByPage：根据给定的页码和每页信息数量，从数据库中检索指定页的用户数据。

rowCount：检索数据库中用户记录的总数。

del：根据给定的 id 从数据库中删除一个用户记录。

getOne：根据给定的 id 从数据库中检索单个用户记录。

save：使用对象的当前值更新数据库中的用户记录。

query：根据可选的过滤条件（如 id、用户名和电子邮件）从数据库中检索用户记录列表。

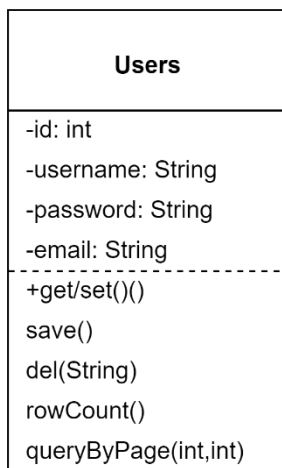


图 4.7 Users 实体类图

#### (2)Groups 类

该类表示一个订阅分组，具有 id：分组的唯一标识符，类型为 int，name：组的名称，类型为 String，userid：分组所属的用户的唯一标识符，类型为 int，created\_at：组创建的日期和时间，类型为 java.util.Date 等属性，以及相应的 getter 和 setter 方法来访问和修改这些字段。

该类中的其他方法包括：

del：删除分组。

getGroups(int userid)：根据用户 id 获取相应分组的列表。

addgroup(Groups gop, int userid)：根据用户 id 添加分组。

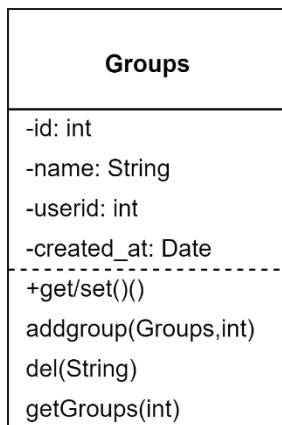


图 4.8 Groups 实体类图

### (3)Subscribe 类

它包含了私有字段用于存储订阅的 ID、URL、名称、描述、所属组 ID、用户 ID 和大小，以及相应的 getter 和 setter 方法来访问和修改这些字段。

除此之外，“Subscribe”类还包含了多个方法，用于从数据库中查询、添加、删除和更新订阅数据。这些方法包括：

list：从数据库中根据给定的组 ID 和用户 ID 查询订阅列表。

add：向数据库中添加一个订阅。

del：根据给定的 ID 从数据库中删除一个订阅。

getOne：根据给定的 ID 从数据库中检索一个订阅。

save：使用对象的当前值更新数据库中的订阅记录。

query：根据可选的过滤条件（如 ID、名称和 URL）从数据库中检索订阅记录列表。

queryByPage：实现了订阅数据的分页查询。

rowCount：返回数据库中订阅记录的总数。

getUrlList：返回用户 ID 的所有订阅的 ID 和 URL 列表。

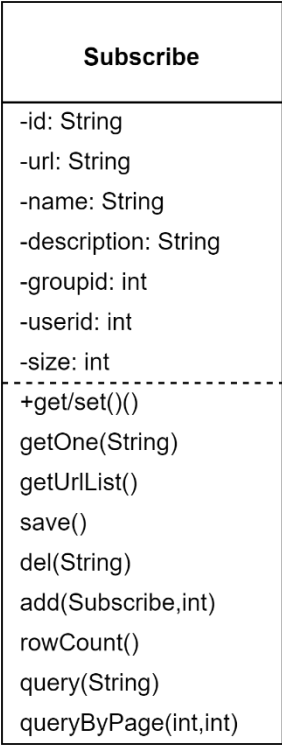


图 4.9 Subscribe 实体类图

(4)Article 类

Article 类有多个属性，例如 id、标题、内容、发布日期、链接、订阅源 id、阅读次数、用户 id、收藏状态、更新日期、作者、是否已阅读和是否已收藏。以及相应的 getter 和 setter 方法来访问和修改这些字段。

该类中的其他方法包括：

- saveArticle()：将新文章保存到数据库中，并返回插入的文章数量。
- articlelist()：从 RSS 源中获取文章列表并返回。
- getSubid()：辅助方法，根据给定的 RSS URL 检索订阅 ID。
- list()：根据特定的订阅源和用户 ID，返回来自数据库的文章列表。
- updateFavorite()：更新数据库中一篇文章的收藏状态。
- favoritelist()：返回当前用户 ID 收藏的文章列表。
- readlist()：返回当前用户 ID 已读的文章列表。
- updateRead()：更新数据库中一篇文章的阅读状态。

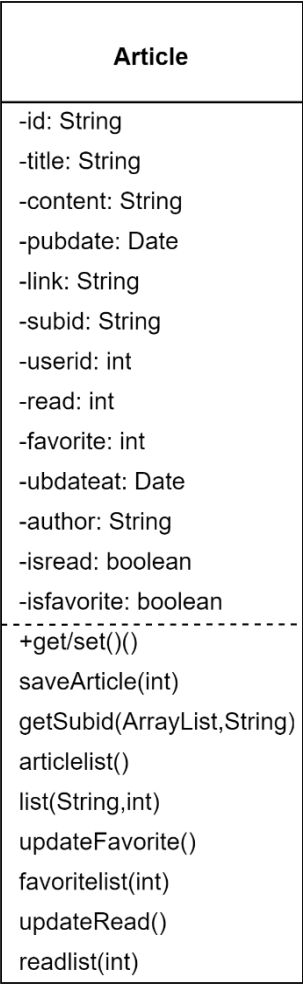


图 4.10 Article 实体类图

4.5.2 实体类之间的关系

实体类和 dehelp 类之间关联，控制器类（Controller Class）调用模型（Model）中各实体类的各种方法。UsersController 类登录过程中会调用密码加密工具 PasswordUtil 类和会话工具 SessionUtil 类，测试 RSS 源和更新文章时 ItemsController 类会调用 RSS 解析工具 RreadRss 类。

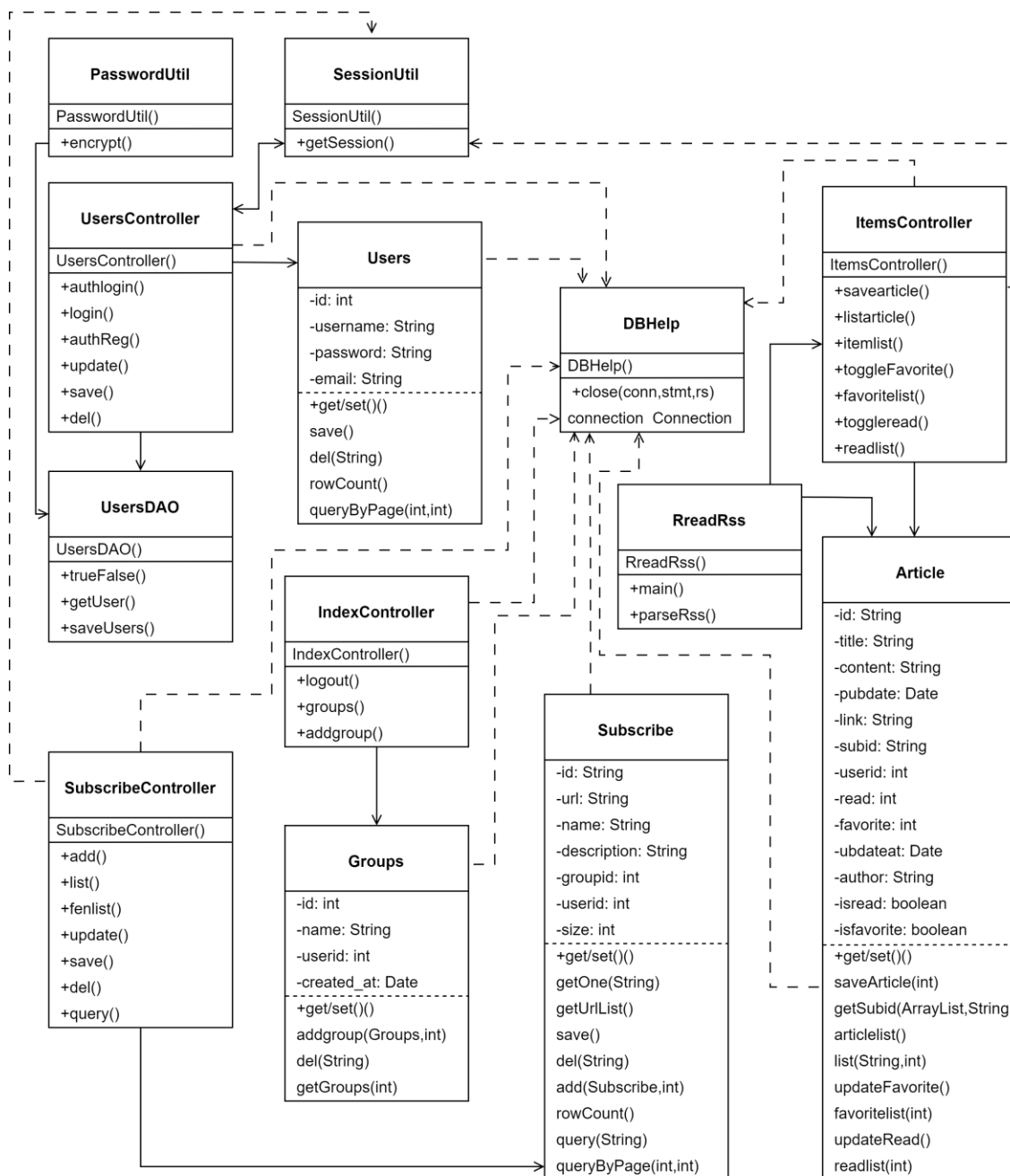


图 4.11 实体类关系图

## 4.6 系统活动图设计

用户在使用本 RSS 阅读器的过程中，使用到的功能可以归纳为以下典型的活动，用户可以在登录后看到自己的分组，可以在主页添加分组和更新文章，可以查看文章列表、管理 RSS 源、和使用用户中心的查看收藏、阅读历史和活跃用户。

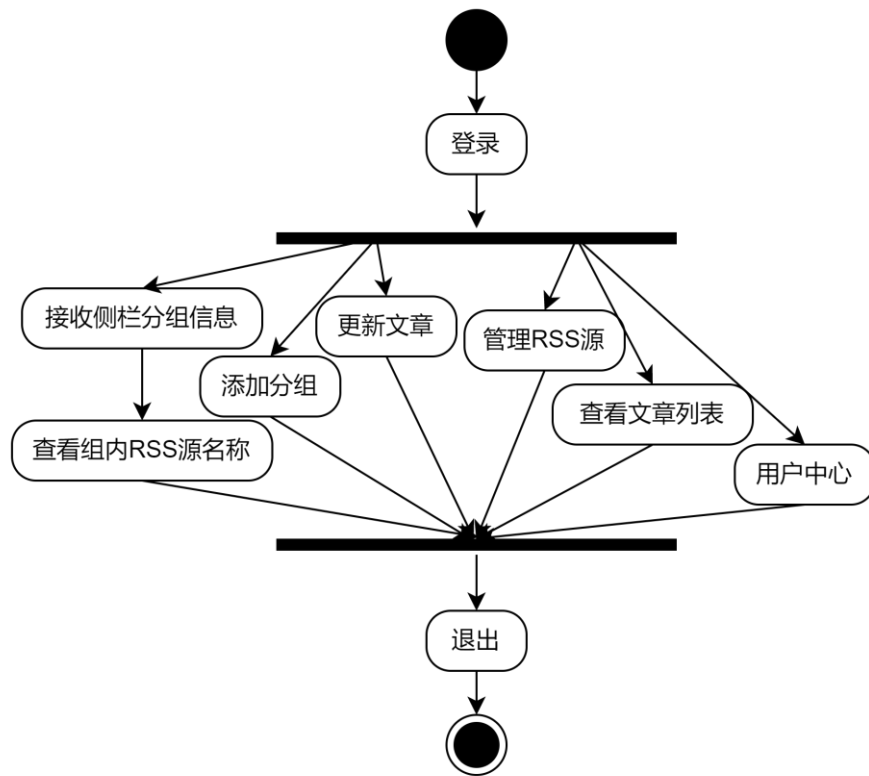


图 4.12 用户活动总图

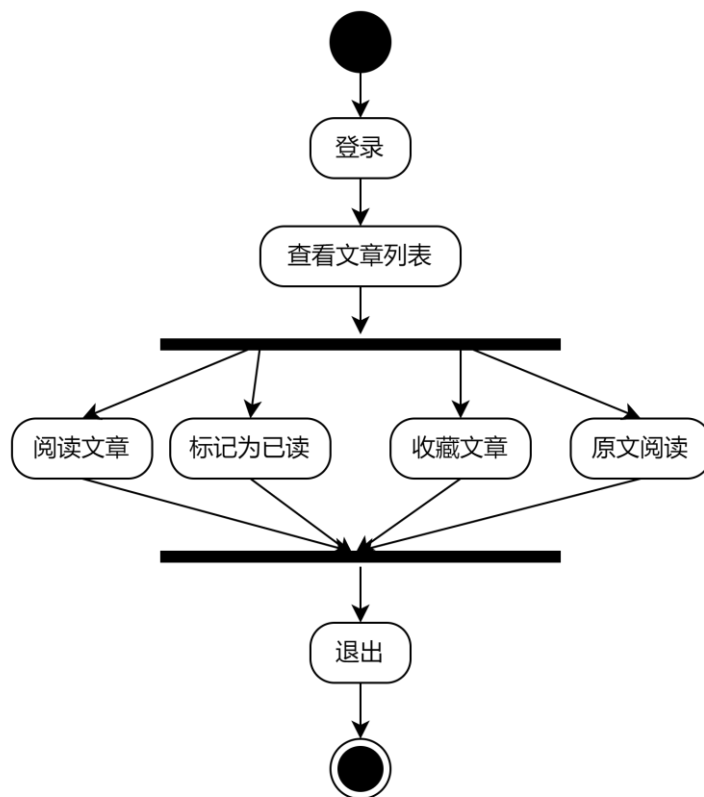


图 4.13 用户管理 RSS 源活动图

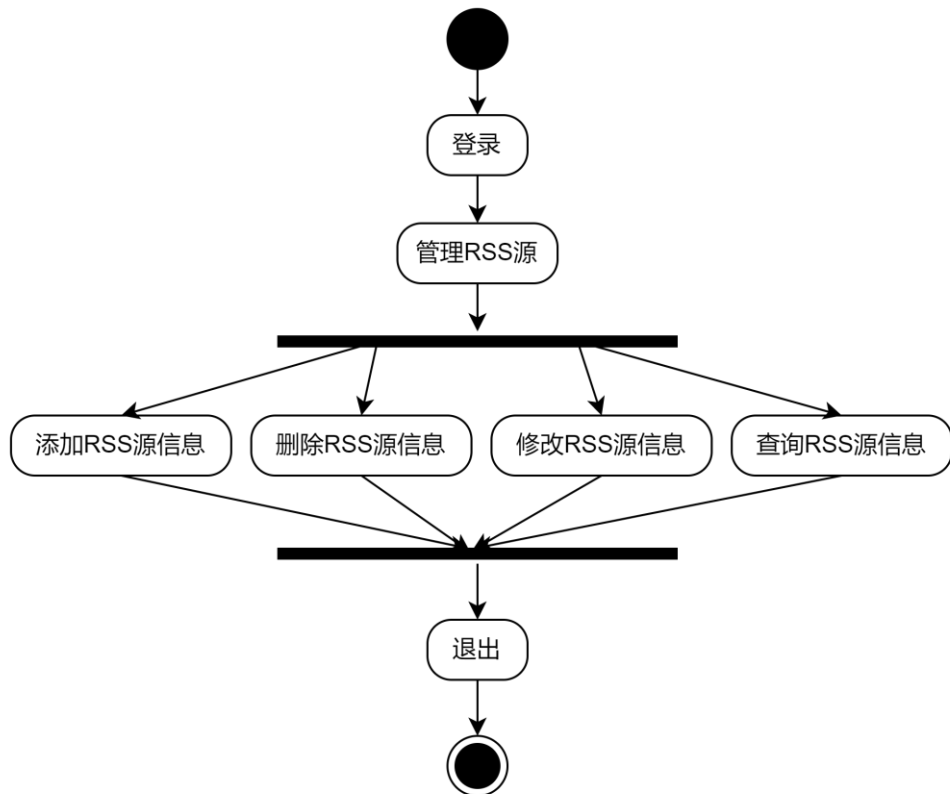


图 4.14 用户查看文章列表活动图

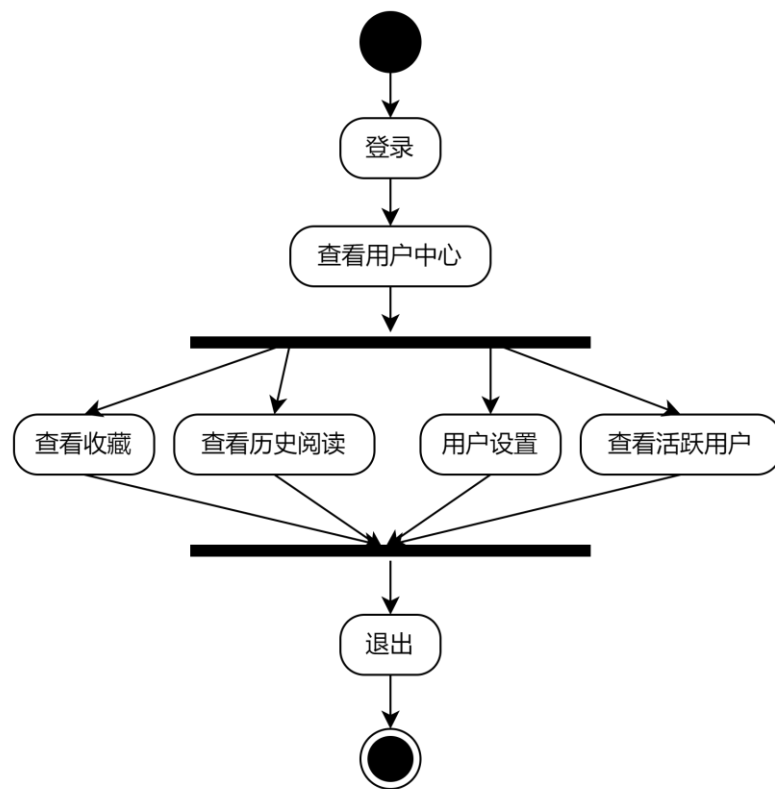


图 4.15 使用用户中心活动图

## 5 系统实现

### 5.1 工具模块

#### 5.1.1 数据库连接模块

公共的数据库连接类 DBHelp 可以在整个应用程序中被多次调用，避免了在每个需要连接数据库的类中都重复编写相同的代码，从而提高了代码重用性和效率。本系统使用 Oracle 数据库用户名为 test，主要的功能代码为：

//获取 connection 对象

```
public static Connection getConnection() {  
    Connection conn = null;  
    try {  
        conn = DriverManager.getConnection(url, user, passWord);  
    } catch (SQLException e) {  
        e.printStackTrace();  
    }  
    return conn;  
}
```

#### 5.1.2 用户密码加密模块

在软件项目开发中，用户密码加密是非常重要的一个环节，它可以保护用户的账户安全。而 PasswordUtil 类就是一个用于密码加密的工具类。该类采用了一种安全可靠的加密算法，可以将用户输入的密码加密后存储到数据库中，避免了密码被恶意获取的风险。此外，当用户登录时调用该类，可以验证用户输入的密码与存储在数据库中的密码是否一致，从而确保用户输入的密码正确性。核心的代码如下：

```
public static String encrypt(String password) {  
    try {  
        MessageDigest md = MessageDigest.getInstance("SHA-256");  
        md.update(password.getBytes());  
        byte[] digest = md.digest();  
        StringBuilder sb = new StringBuilder();  
        for (byte b : digest) {  
            sb.append(String.format("%02x", b & 0xff));  
        }  
        return sb.toString();  
    } catch (NoSuchAlgorithmException e) {  
        throw new RuntimeException("加密算法不存在", e);  
    }  
}
```

#### 5.1.3 会话工具模块

RSS 阅读器的每位用户的数据应当独立，应实现当前登录用户只能操作自己的数据，使用 SessionUtil 类可以便捷地处理用户的会话信息，这个模块可以方便地获取当前用户的会话属性，比如可以用来检测用户是否已经登录，以及获取已登录用户的信息等。核心代码如下：

```
public class SessionUtil {
    public static Users getSession(HttpServletRequest request) {
        HttpSession session = request.getSession();
        Users user = (Users) session.getAttribute("user");
        return user;
    }
}
```

## 5.2 用户使用功能模块

### 5.2.1 用户登录功能

拾读 RSS 阅读器的用户需要在登录页面 (<http://localhost:8080/rssreaderdemo/login.do>) 输入其账号和密码，然后点击登录。这个过程中前端页面使用 form 表单提交到后端的/authlogin.do 接口。后端 UsersController 控制器中的 authlogin 方法接收到请求后，首先获取用户名和密码，然后会调用 UsersDAO 对象的 getUser 方法，查询数据库中是否存在该用户。如果查询到了该用户，则会将用户信息存储在 HttpSession 中，并使用 ModelAndView 对象返回登录成功的页面；反之，如果未查询到该用户，则设置视图名称为登录失败的页面，并返回 ModelAndView 对象。最终，前端根据后端返回的结果，跳转到对应的页面。核心代码如下：

```
@RequestMapping("/authlogin.do")
public ModelAndView authlogin(
    HttpServletRequest request,
    HttpServletResponse response
) throws UnsupportedEncodingException {
    request.setCharacterEncoding("utf-8");
    ModelAndView mv = new ModelAndView();
    String username = request.getParameter("username"); //取得表单中传递过来的用户名和密码
    String password = request.getParameter("password");
    System.out.println(username); //测试
    UsersDAO userdao = new UsersDAO();
    Users user = userdao.getUser(username, password);
    if (user != null) { //与数据库里匹配，看有没有这个用户名与密码
        HttpSession session = request.getSession();
        session.setAttribute("user", user);
        return new ModelAndView("login_success.html");
    } else {
```

```
mv.setViewName("/failure1.html");
}
return mv;
}
```

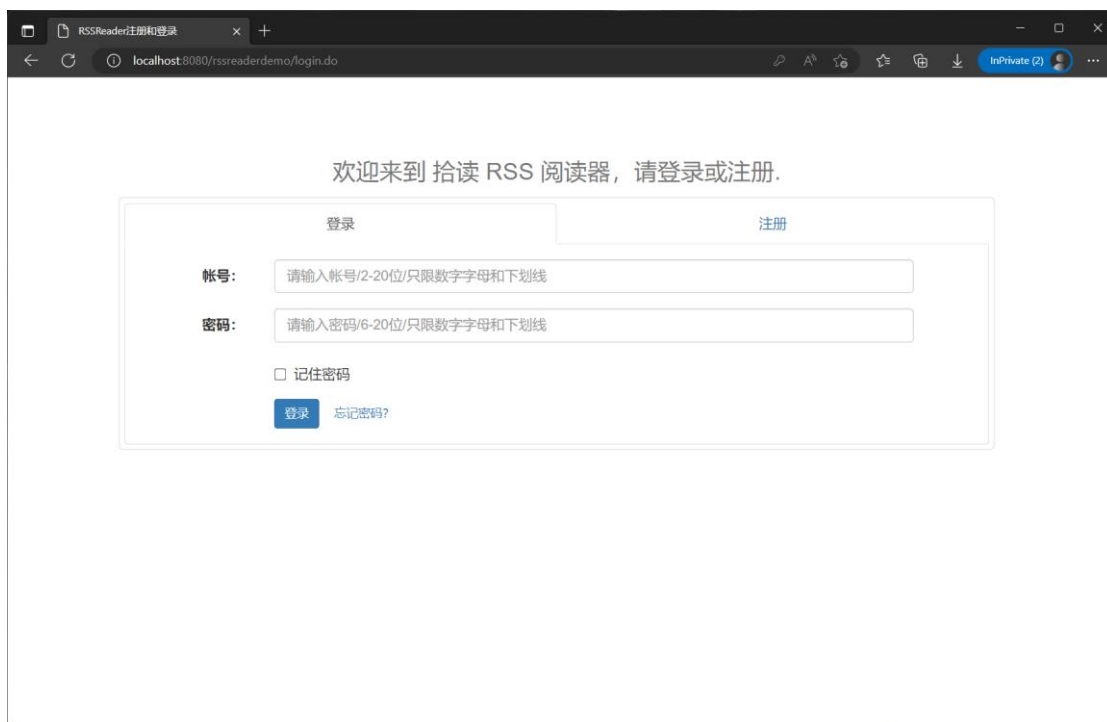


图 5.1 用户登录

### 5.2.2 用户注册功能

在注册时，用户输入用户名、密码、邮箱等信息后，使用 form 表单提交到后端的/authReg.do 接口。后端 UsersController 控制器中的 authReg 方法接收到请求后，首先获取用户信息，然后调用 UsersDAO 对象的 saveUsers 方法将用户信息保存到数据库中。如果保存成功，则设置 ModelAndView 的视图名称为登录页面；如果保存失败，则设置视图名称为注册失败的页面。最终，前端根据后端返回的结果，跳转到对应的页面。核心代码如下：

```
@RequestMapping("/authReg.do")
public ModelAndView authReg(
    HttpServletRequest request,
    Users user,
    Model model
){
    try {
        request.setCharacterEncoding("UTF-8");
    } catch (UnsupportedEncodingException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
```

```
}  
ModelAndView mv = new ModelAndView();  
UsersDAO usersdao = new UsersDAO();  
if (usersdao.saveUsers(user) != 0) {  
    mv.setViewName("login.do");  
} else {  
    mv.setViewName("failure1.html");  
}  
return mv;  
}
```

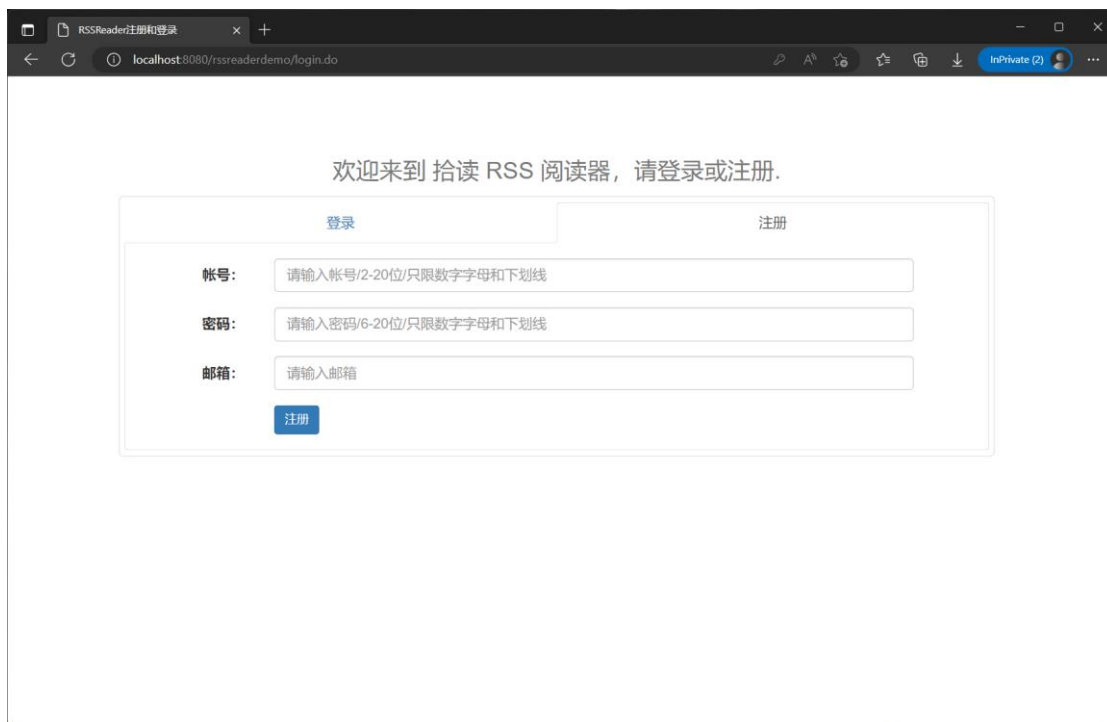


图 5.2 用户注册

### 5.2.3 用户退出登录

当用户在主界面点击左上方的“当前用户”下的退出按钮时，前端会向后端发送请求，要求退出登录。后端会接收到该请求并执行以下操作：首先，获取当前 session 中保存的用户信息，并将其从 session 中移除；接着，返回重定向到登录页面的结果，从而实现退出登录的功能。最后，用户会被重定向到用户登录界面。

核心代码如下：

```
@RequestMapping("/logout")  
public String logout(HttpServletRequest request) {  
    // 注销当前登录用户的信息  
    HttpSession session = request.getSession();  
    session.removeAttribute("user");  
    // 注销成功，跳转到登录页面  
    return "redirect:/login.do";  
}
```

}

5.2.4 用户功能主界面

拾读的主页允许用户添加 RSS 源的分组，并在需要阅读的时候更新文章，方便用户随时了解最新动态。用户可以创建新的分组，可以在管理 RSS 源的时候将不同的 RSS 源添加到不同的分组中。当用户刷新拾读的主页时可以看到侧边栏的分组。



图 5.3 用户功能主界面

当用户需要创建分组时，在页面中，用户输入分组名称和相关信息后，提交表单到后端的/addgroup.do 接口。后端 addgroup 方法会接收到请求并执行以下操作，首先获取分组信息，然后调用 Groups 对象的 addgroup 方法将分组信息保存到数据库中。最终，后端返回重定向到欢迎页面的结果，实现添加分组的功能。核心代码如下：

```
@RequestMapping("/addgroup.do")
public String addgroup(Groups gop, HttpServletRequest request)
    throws SQLException {
    //调用模型的添加方法
    int userid = SessionUtil.getSession(request).getId();
    Groups group = new Groups();
    group.addgroup(gop, userid);
    return "welcome.html";
}
```

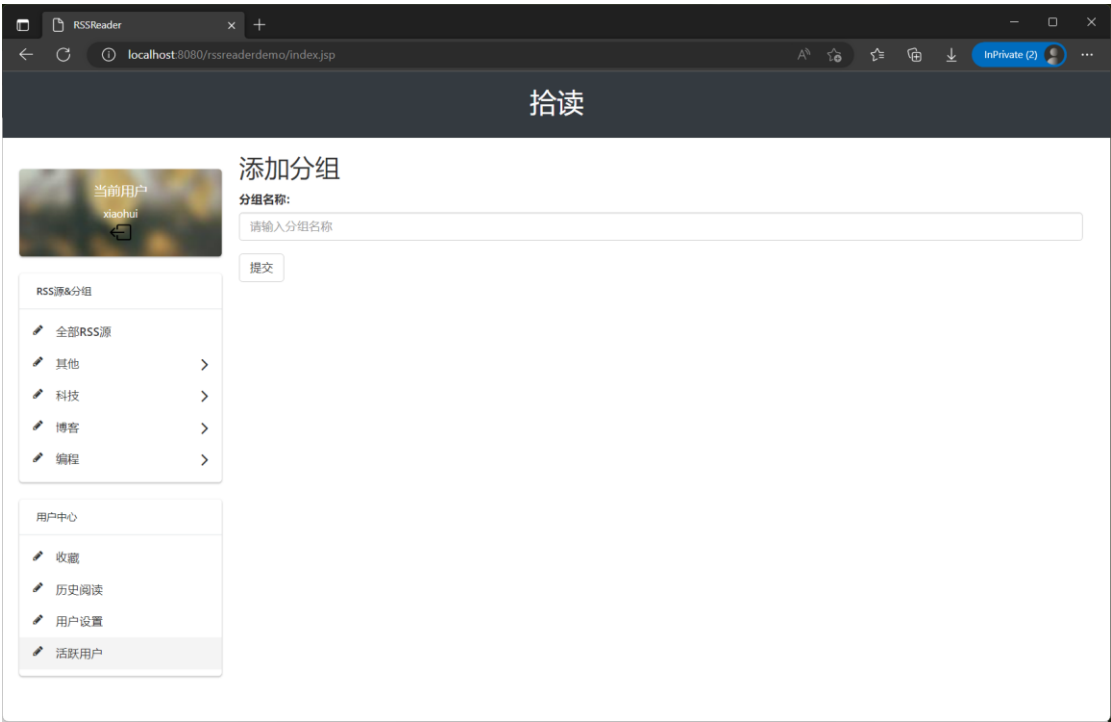


图 5.4 添加分组

5.2.5 RSS 源管理

用户可以借助拾读提供的 RSS 源管理功能，轻松地添加、删除、修改和查询 RSS 源。为了添加新的 RSS 源，用户需要提供该 RSS 源的 URL 地址。在删除或修改 RSS 源时，用户可以在管理页面中找到相应的选项。

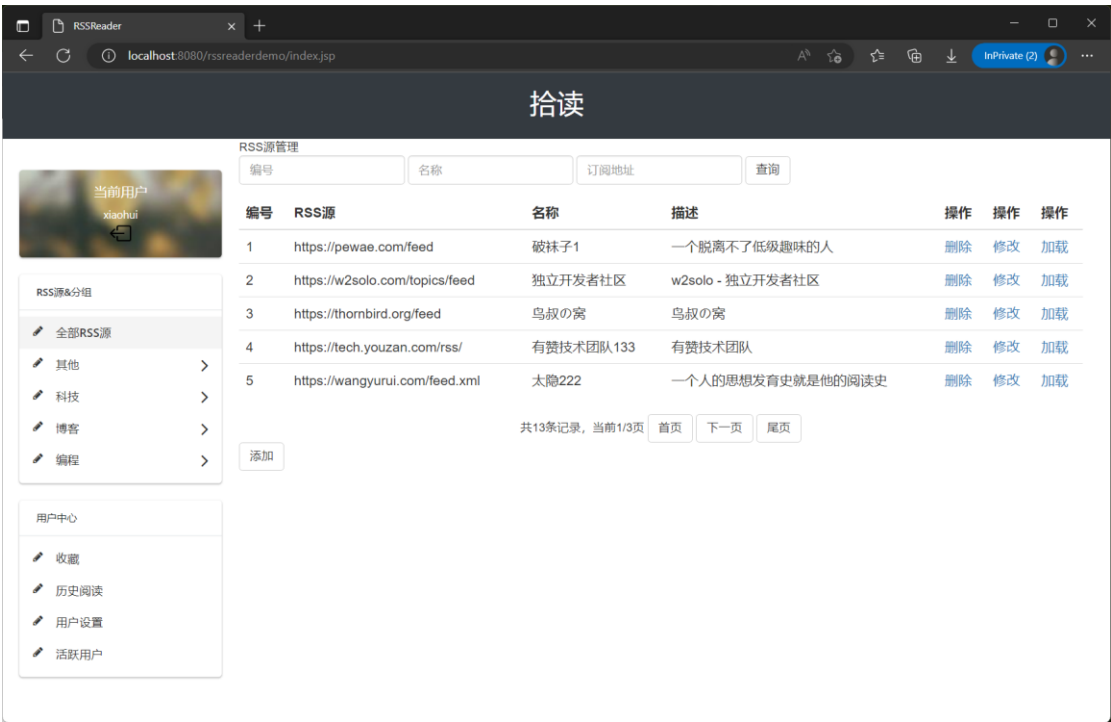


图 5.5 RSS 源管理

在前端 RSS 源管理页面中，用户选择要修改编号为 2 的 RSS 源信息后，使用修改链接向后端的 /update.do 接口发送请求。后端 update 方法接收到请求后，首先获取要修改的 RSS 源信息的 ID，然后调用 Subscribe 对象的 getOne 方法根据 ID 获取要修改的 RSS 源信息，并将其添加到 Model 中，返回 sub\_update.jsp 视图。用户在 sub\_update.jsp 页面中修改 RSS 源信息并提交表单。前端使用 form 表单提交到后端的 /save.do 接口，后端 save 方法接收到请求后，调用 Subscribe 对象的 save 方法保存修改后的 RSS 源信息。最终，后端返回重定向到 RSS 源管理页面的结果，实现修改 RSS 源信息的功能。核心代码如下：

```
@RequestMapping("/update.do")
public String update(String id, Model model) throws SQLException {
    //根据 id 获得一个对象
    Subscribe sub = new Subscribe().getOne(id);
    //把这个对象传递给视图显示
    model.addAttribute("sub", sub);
    return "sub_update.jsp";
}

@RequestMapping("/save.do")
public String save(Subscribe sub) throws SQLException {
    //调用模型的 save 方法
    sub.save();
    return "fenlist.do";
}
```

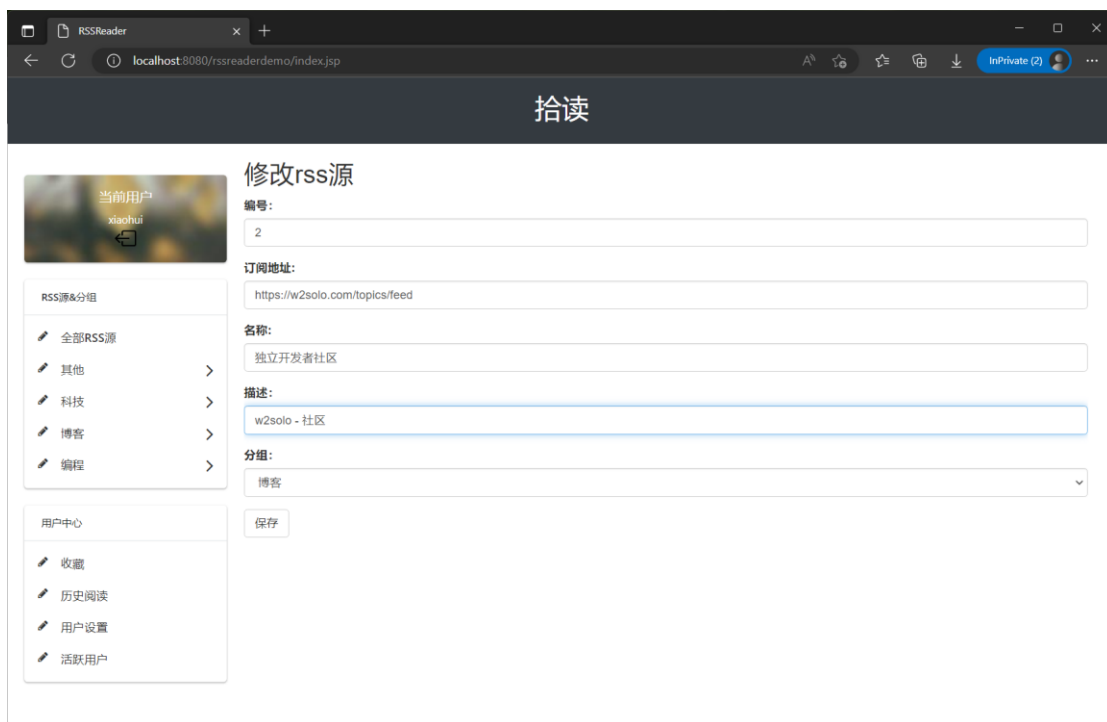


图 5.6 修改 RSS 源

点击保存，修改后的内容会保存到数据库。

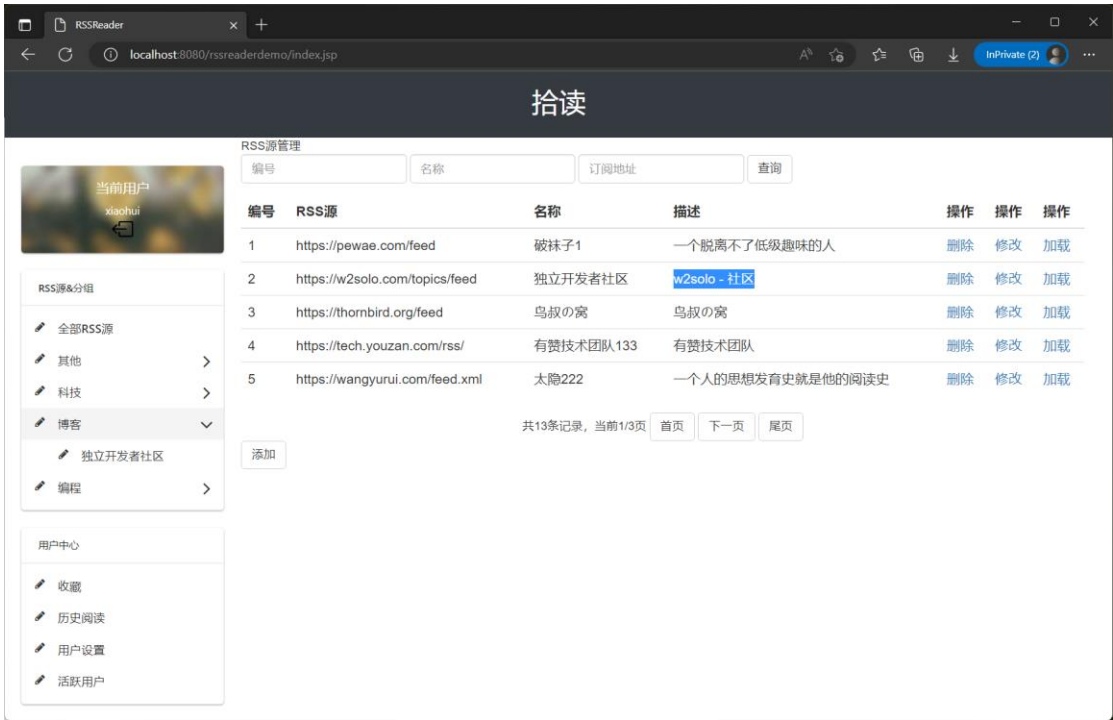


图 5.7 修改 RSS 源保存后

在前端 RSS 源管理页面中，用户选择编号为 1 的 RSS 源信息后，点击删除操作向后端的 /del.do 接口发送请求。后端 del 方法接收到请求后，首先获取要删除的 RSS 源信息的 ID，然后调用 Subscribe 对象的 del 方法根据 ID 删除 RSS 源信息。最终，后端返回重定向到分组列表页面的结果，实现删除 RSS 源信息的功能。核心代码如下：

```
@RequestMapping("/del.do")
public String del(String id) throws SQLException {
    //调用模型的删除方法
    Subscribe subscribe = new Subscribe();
    subscribe.del(id);
    return "fenlist.do";
}
```

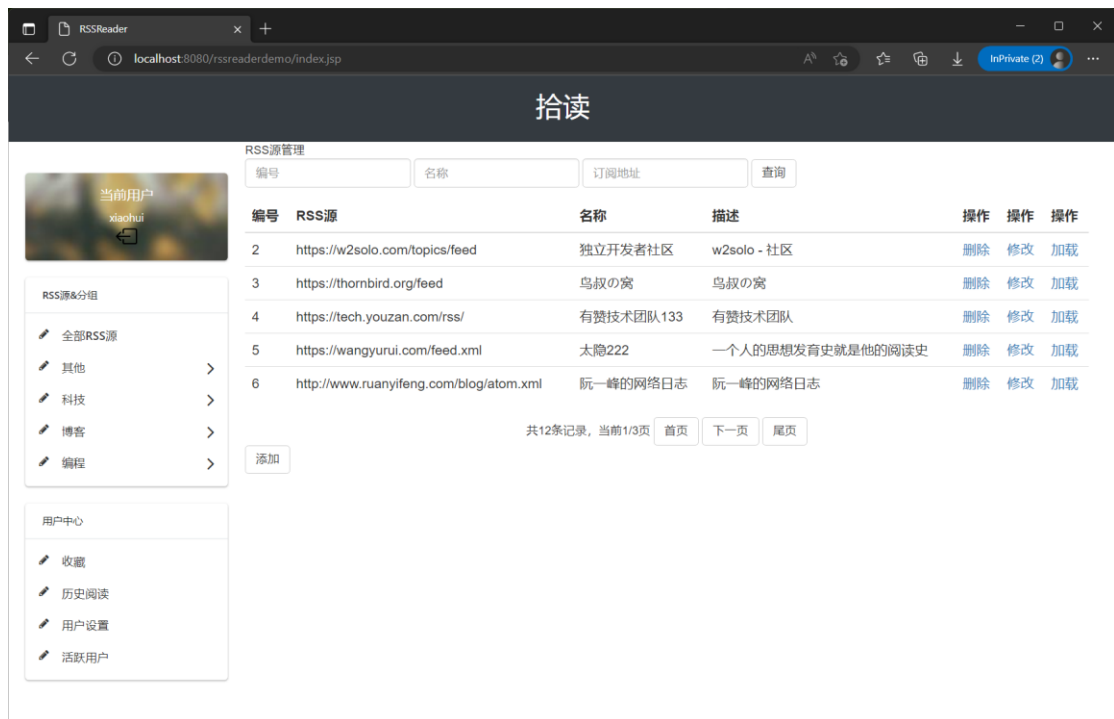


图 5.8 删除编号为 1 的 RSS 源后

添加一个新的 RSS 订阅源，分组选择其他，用户输入 RSS 源的相关信息后，使用 form 表单提交到后端的/add.do 接口。后端 add 方法接收到请求后，首先获取 RSS 源信息，然后调用 Subscribe 对象的 add 方法将 RSS 源信息保存到数据库中。最终，后端返回重定向到 RSS 管理列表页面的结果，实现添加 RSS 源信息的功能。核心代码如下：

```
@RequestMapping("/add.do")
public String add(Subscribe sub, HttpServletRequest request)
    throws SQLException {
    int userid = SessionUtil.getSession(request).getId();
    //调用模型的添加方法
    Subscribe subscribe = new Subscribe();
    subscribe.add(sub, userid);
    return "fenlist.do";
}
```

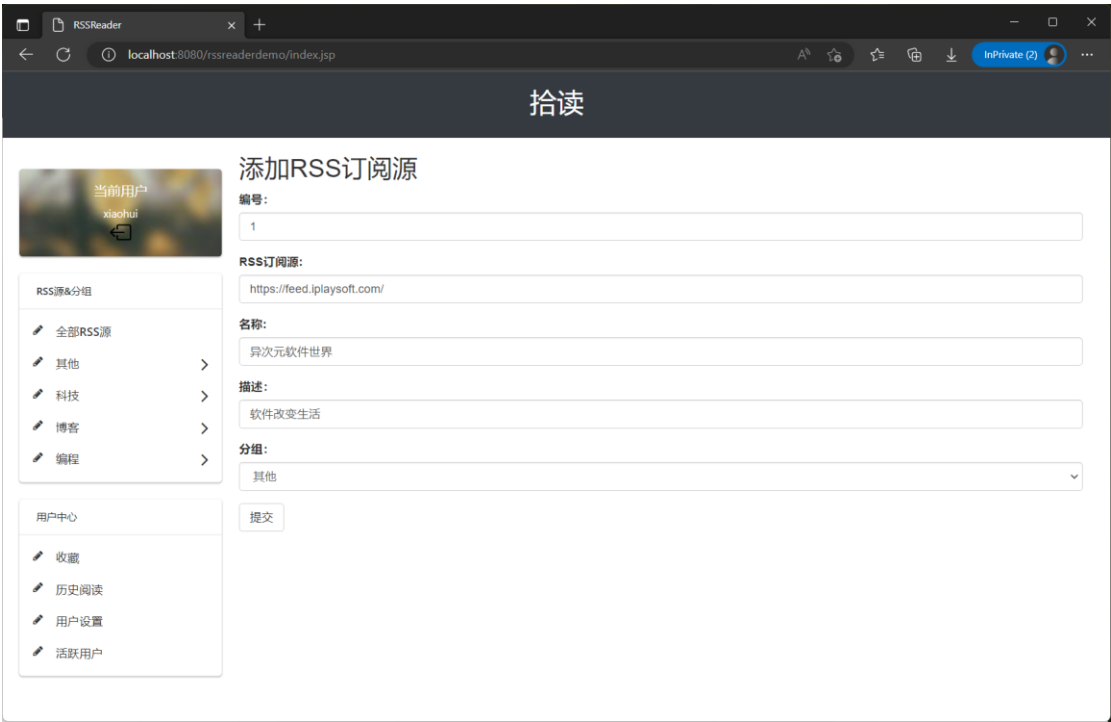


图 5.9 添加新的 RSS 源

添加 RSS 信息后，用户需要点击提交按钮将其保存到数据库中。保存成功后，这些信息将被显示在 RSS 源管理界面上。

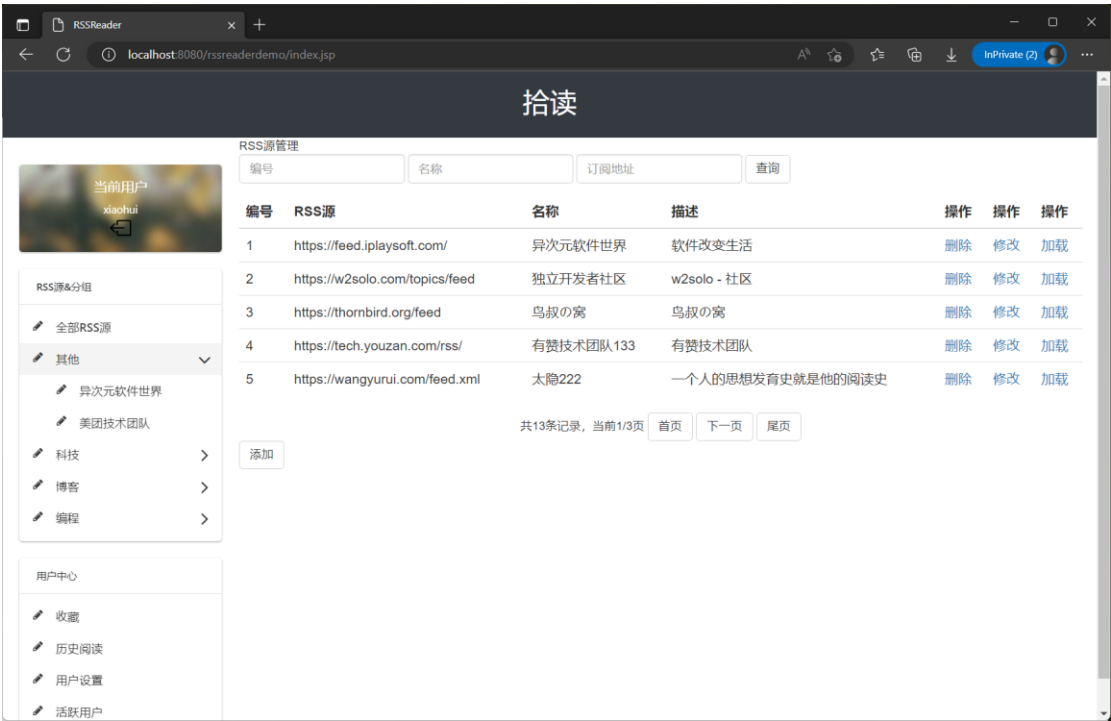


图 5.10 添加新的 RSS 源后

在前端页面中，用户输入 RSS 源地址后，点击每则 RSS 信息后的加载操作可以请求后端的/itemlist.do 接口。后端 itemlist 方法接收到请求后，首先获取 RSS

源地址，然后调用 RreadRss 对象的 parseRss 方法解析 RSS 源地址。如果解析成功，将解析结果存放到 SyndFeed 对象中，并将 SyndFeed 对象和 entriesList 添加到 Model 中，返回 items\_test.jsp 视图。如果解析失败，返回 rssurl\_error.jsp 视图，提示用户 RSS 源地址无效。这个过程主要用于测试 RSS 源地址是否正确以及解析 RSS 源是否成功，并不会将文章存储到数据库中，方便用户测试 RSS 源是否有效。核心代码如下：

```
@SuppressWarnings("null")
@RequestMapping("/itemlist.do")
public String itemlist(String rss, Model model) throws SQLException {
    //调用 RreadRss 方法
    //获取基于 rome 的解析对象
    RreadRss rreadrss = new RreadRss();
    //解析 rss 内容，存放到 SyndFeed 对象中
    SyndFeed feed = rreadrss.parseRss(rss);
    if (feed == null) return "rssurl_error.jsp";
    List entriesList = feed.getEntries();
    model.addAttribute("feed", feed);
    model.addAttribute("entriesList", entriesList);
    return "items_test.jsp";
}
```



图 5.11 测试 RSS 源的有效性 a

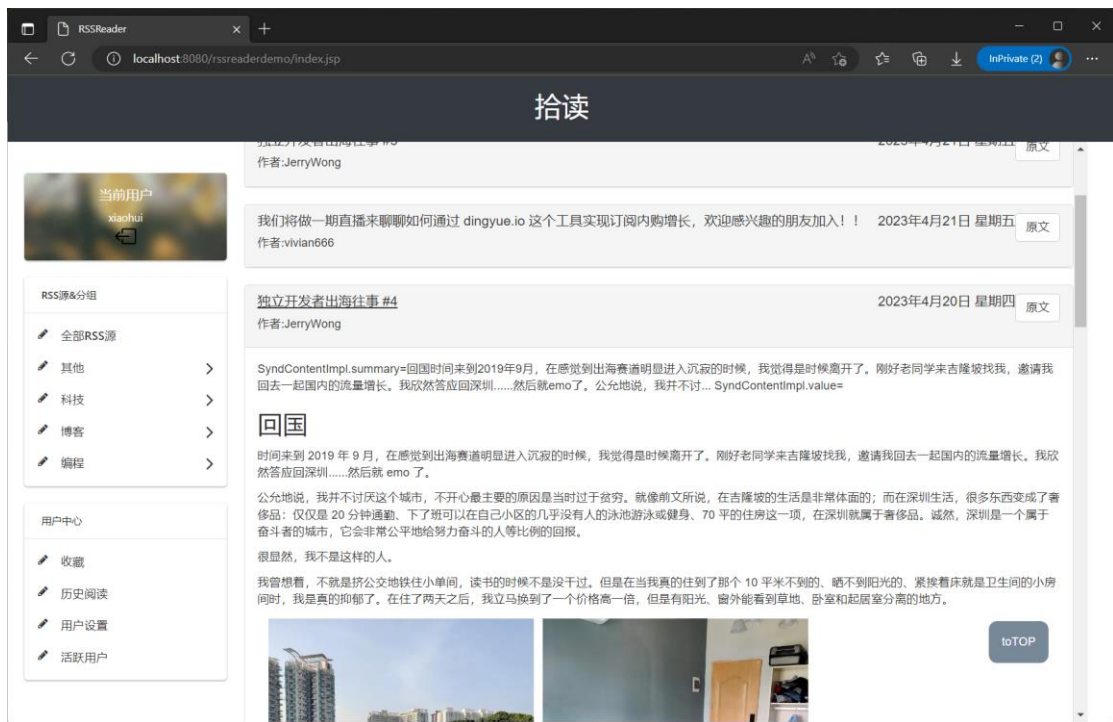


图 5.12 测试 RSS 源的有效性 b

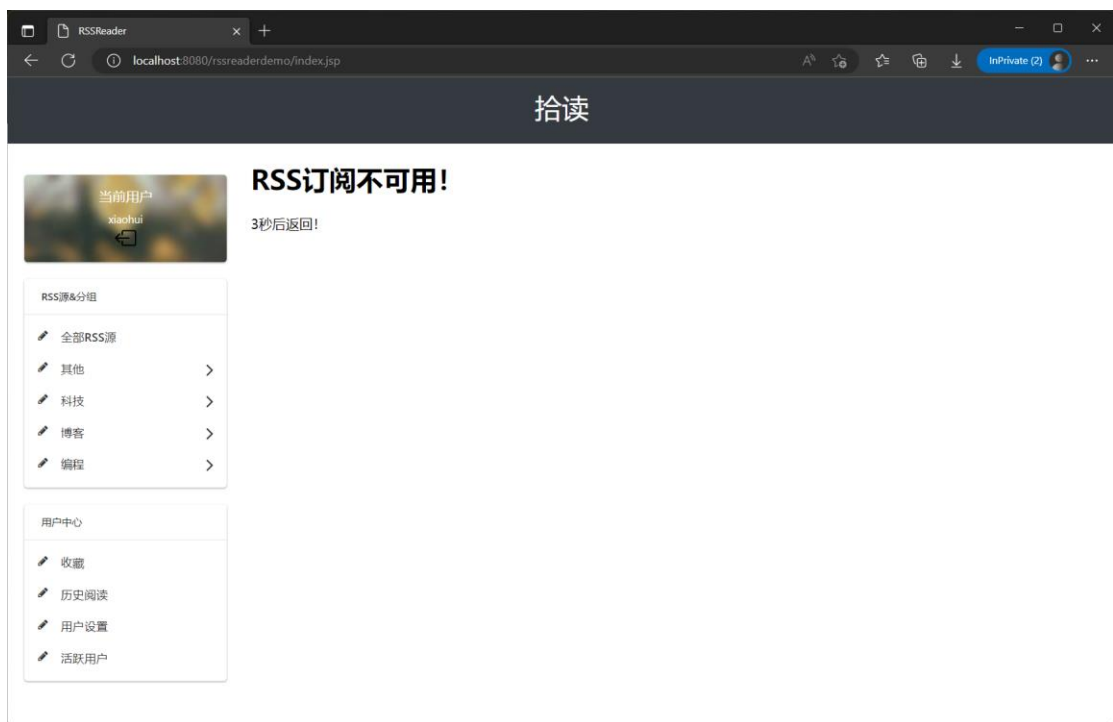


图 5.13 测试 RSS 源的有效性 c

在 RSS 源管理界面上可以对 RSS 源信息进行组合查询。在前端 RSS 源管理页面中，用户输入查询条件后，使用 form 表单提交到后端的 /query.do 接口。后端 query 方法接收到请求后，首先获取用户输入的查询条件，然后调用 Subscribe 对象的 query 方法查询符合条件的 RSS 源信息，并将查询结果添加到 Model 中，返回

sub\_query.jsp 视图，显示查询结果。这个过程主要用于方便用户快速查找符合条件的 RSS 源信息。核心代码如下：

```
@RequestMapping("/query.do")
public String query(Model model, String id, String name, String url)
    throws SQLException {
    //调用模型的 list 方法
    Subscribe sub = new Subscribe();
    ArrayList list = sub.query(id, name, url);
    //结果传递给视图 list.jsp 显示
    model.addAttribute("list", list);
    return "sub_query.jsp";
}
```

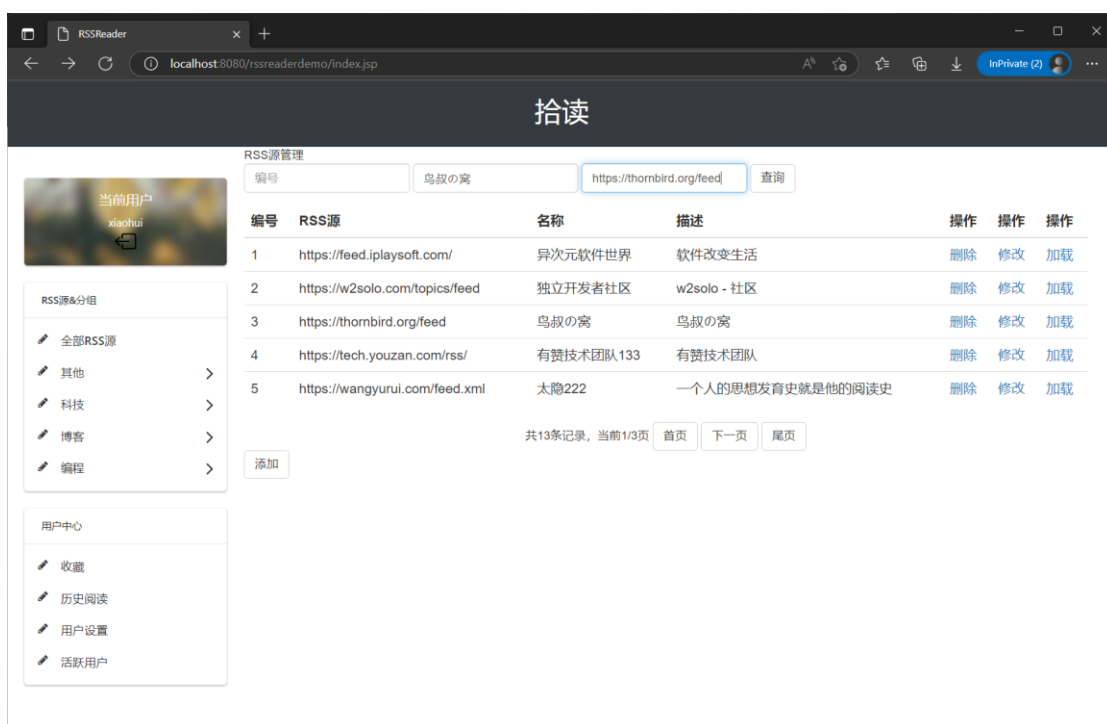


图 5.14 查询 RSS 源信息

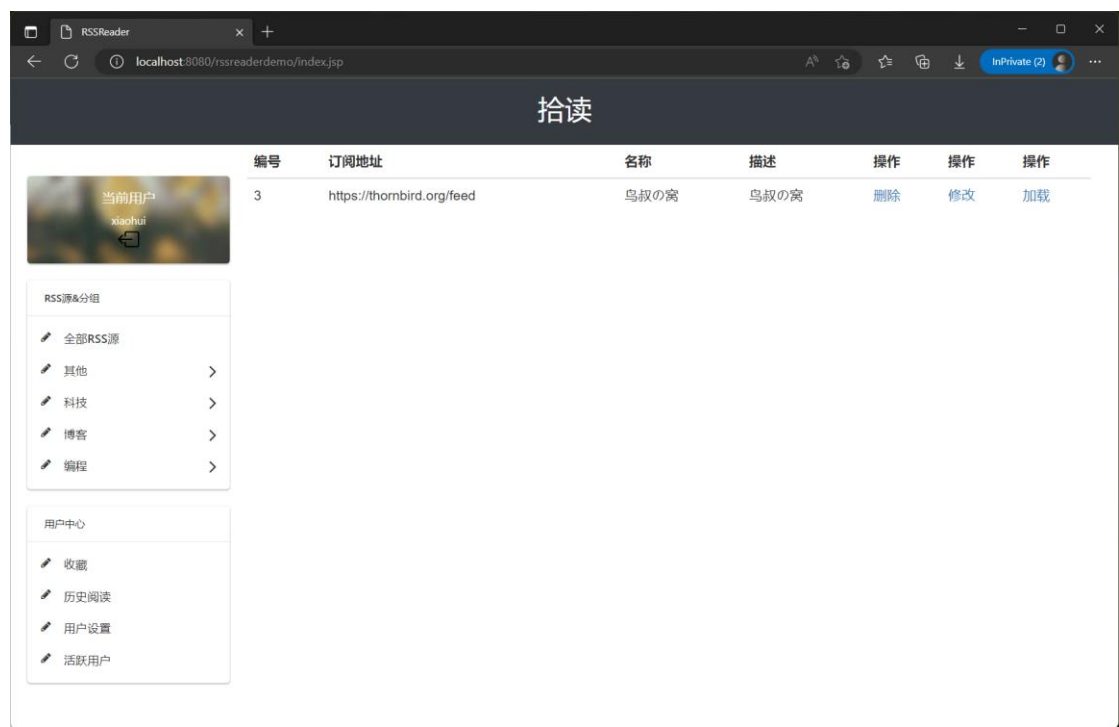


图 5.15 查询 RSS 源信息成功

RSS 源管理功能界面的信息分页显示，可以点击下方功能按钮进行跳转，分页功能的实现过程如下：

首先获取总记录数 totalRow，并设置每页显示的记录数 pageSize。

根据 totalRow 和 pageSize 计算出总页数 totalPage。

获取当前页号 pageNo，如果没有指定，默认为第一页。

调用 Subscribe 对象的 queryByPage 方法，获取指定页的记录列表 list。

在页面上，会显示分页导航栏，其中包含了链接到首页、上一页、下一页和尾页的按钮，以及当前页号和总页数的信息。

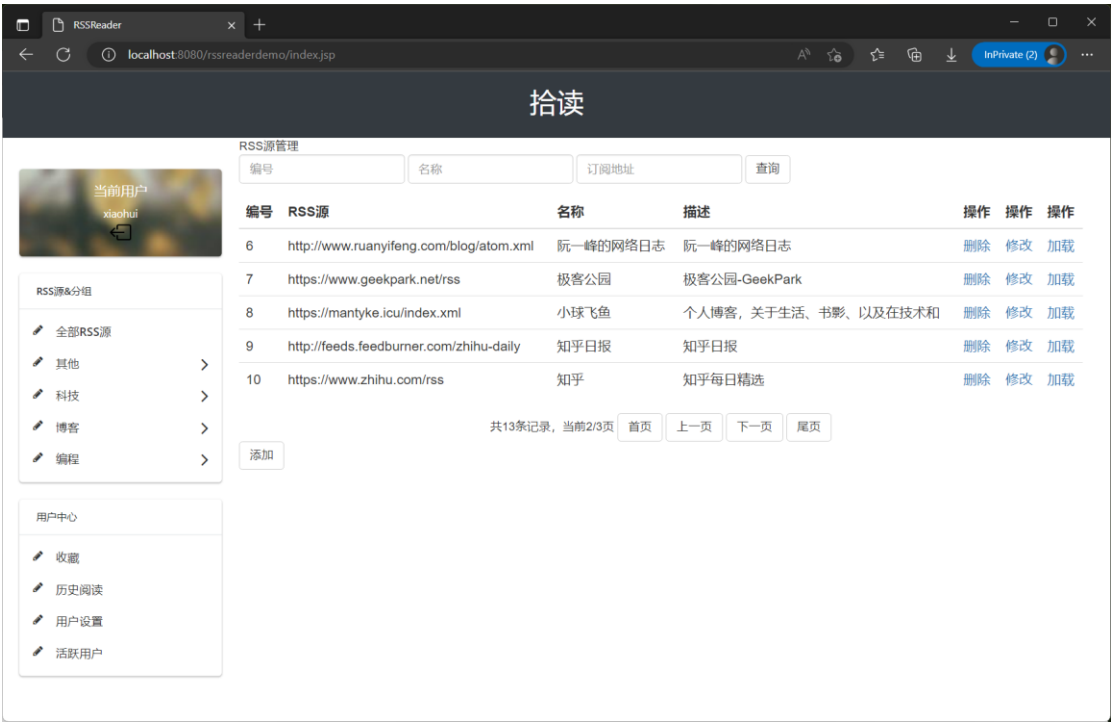


图 5.16 点击下一页后

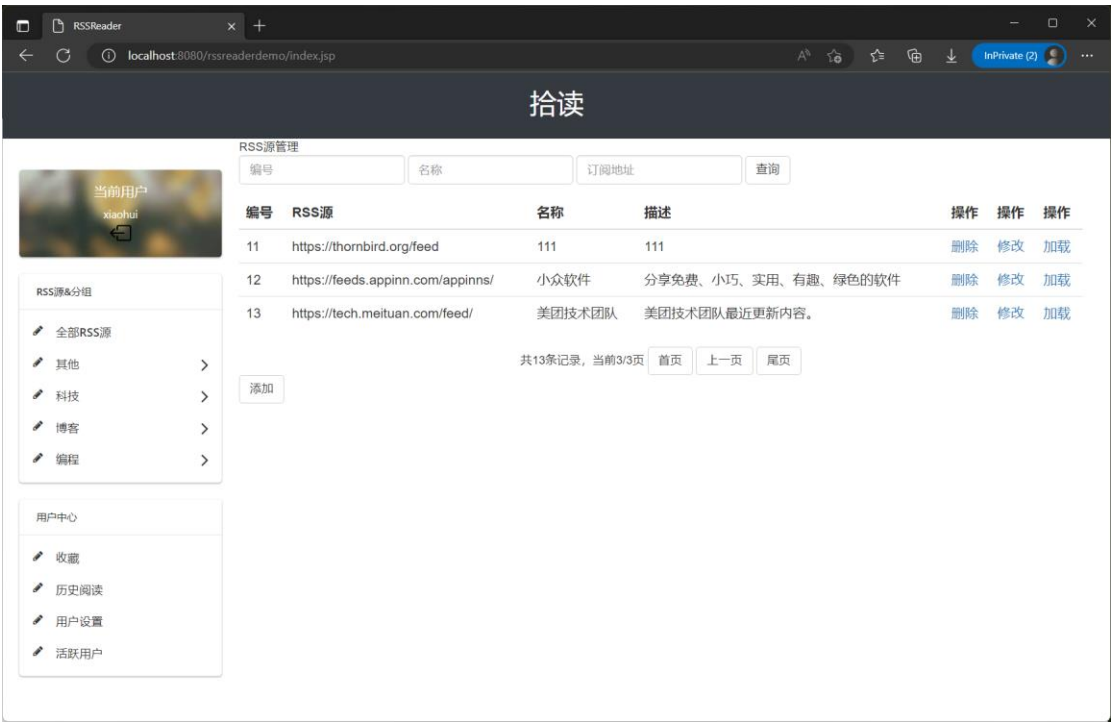


图 5.17 再次下一页或点击尾页后

5.2.6 文章列表

在点击侧边栏分组内的 RSS 源名称后，用户可以轻松查看文章列表。当用户阅读文章时，可以使用拾读的工具栏对文章进行标记和收藏，并通过单击相应按钮来跳转到原文。

在前端页面中，用户点击侧边栏的 RSS 源名称后发送信息给后端的 /listarticle.do 接口。后端 listarticle 方法接收到信息后，首先获取用户选择的 RSS 源 ID，然后调用 Article 对象的 list 方法查询该 RSS 源的文章列表，并将查询结果添加到 Model 中，返回 article.jsp 视图，显示查询结果。这个过程主要用于让用户快速查看指定 RSS 源的文章列表内容。核心代码如下：

```
@RequestMapping("/listarticle.do")
public String listarticle(
    String subid,
    Model model,
    HttpServletRequest request
) throws SQLException {
    Article article = new Article();
    int userid = SessionUtil.getSession(request).getId();
    ArrayList<Article> articlelist = article.list(subid, userid);
    model.addAttribute("articlelist", articlelist);
    return "article.jsp";
}
```

用户点击文章收藏图标后，向后端 /toggleFavorite.do 接口发送请求，请求切换指定文章的收藏状态。后端 toggleFavorite 方法接收到请求后，首先获取要切换收藏状态的文章信息，然后调用 Article 对象的 updateFavorite 方法切换指定文章的收藏状态，并根据切换后的状态更新该文章的收藏状态。最后，重定向到 /listarticle.do 路径，将用户返回到指定 RSS 源的文章列表页面，并显示更新后的文章收藏状态。核心代码如下：

```
@RequestMapping("/toggleFavorite.do")
public String toggleFavorite(Article article) throws SQLException {
    // 切换收藏状态
    String sub_id = article.getSubid();
    article.updateFavorite();
    return "redirect:/listarticle.do?subid=" + sub_id;
}
```

用户点击文章标记已读未读的图标后，向后端 /toggleread.do 接口发送请求，请求切换指定文章的已读状态。后端 toggleread 方法接收到请求后，首先获取要切换已读状态的文章信息，然后调用 Article 对象的 updateRead 方法切换指定文章的已读状态，并根据切换后的状态更新该文章的已读状态。最后，返回到指定 RSS 源的文章列表页面，重新加载文章列表并显示更新后的已读状态。核心代码如下：

```
@RequestMapping("/toggleread.do")
public String toggleread(Model model, Article article) throws SQLException {
```

```
// 切换收藏状态
article.updateRead();
String sub_id = article.getSubid();
return "redirect:/listarticle.do?subid=" + sub_id;
}
```



图 5.18 文章列表页面

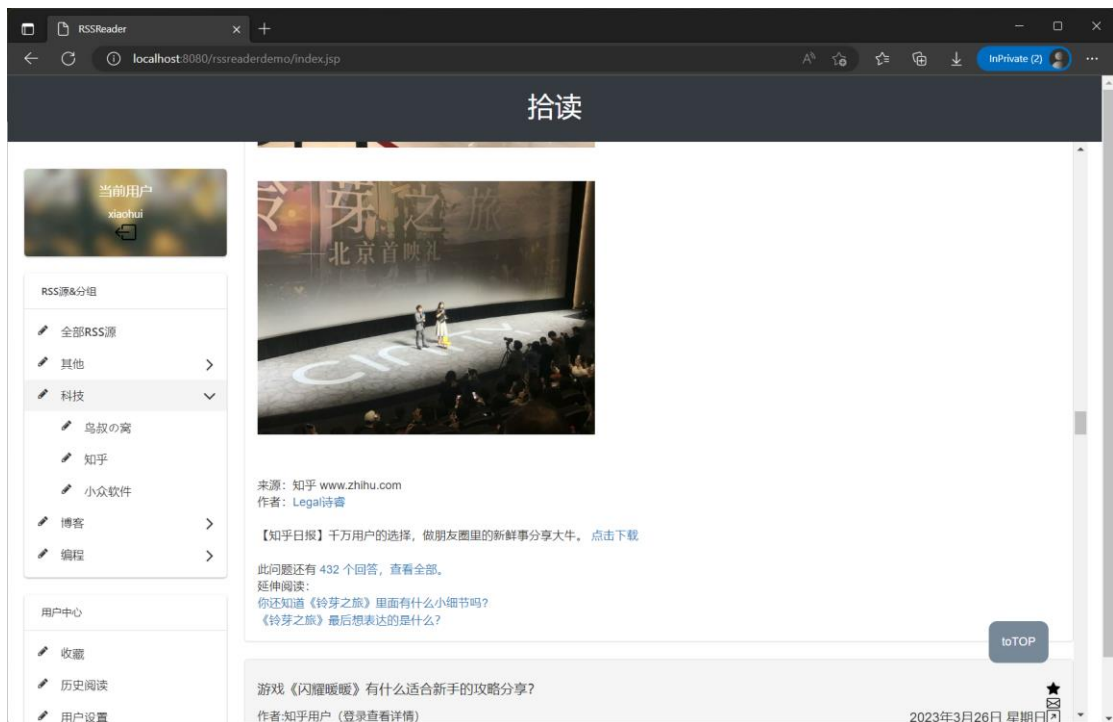


图 5.19 在文章列表页面阅读文章



图 5.20 跳转原文阅读文章

5.2.7 文章收藏历史

拾读允许用户查看收藏历史，并可以在此处阅读或跳转到原文。同时，用户可以取消收藏。在收藏历史页面中，用户可以看到他们以前收藏的所有文章，并可以取消收藏。



图 5.21 文章收藏页面

将第二篇文章取消收藏后



图 5.22 取消收藏第二篇后

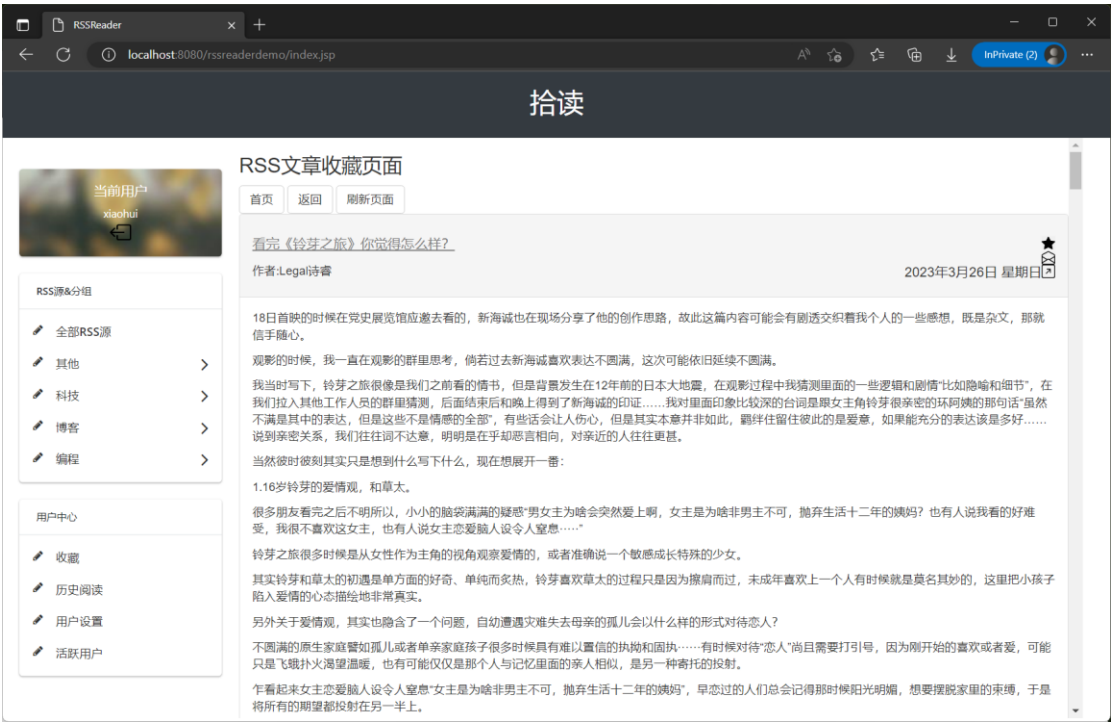


图 5.23 在收藏页面阅读文章

5.2.8 文章阅读历史

拾读提供了查看阅读历史的功能，用户可以在此处查看已读过的文章。用户可以将文章标记为未读，也可以在此处阅读或跳转到原文。在阅读历史页面中，用户

可以看到他们以前阅读过的所有文章，并可以将已读文章标记为未读。



图 5.24 文章阅读历史页面

将上图中第三篇文章标记为未读。



图 5.25 第三篇文章标记为未读后

## 6 系统测试

### 6.1 用户登录功能测试

用户登录功能测试的主要目的是验证用户登录系统的流程是否正常，并确定用户是否能够成功登录系统。在进行用户登录功能测试时，需要考虑多种情况，以确保系统的健壮性和可靠性。

表 6.1 用户登录测试项

| 测试用例                | 测试结果      | 记录结果 |
|---------------------|-----------|------|
| 输入正确的用户名和密码         | 登录成功      | 测试成功 |
| 输入错误的用户名和密码         | 登录失败并显示错误 | 测试成功 |
| 用户名和密码是否有格式限制       | 是         | 测试成功 |
| 登录后，页面是否正确跳转到用户个人主页 | 正确跳转      | 测试成功 |
| 登录后，是否能正常使用网站的其他功能  | 正常使用      | 测试成功 |

### 6.2 用户注册功能测试

用户注册功能测试的主要目的是验证用户注册系统的流程是否正常，并确认用户是否能够成功注册并创建账户。通过对用户注册功能的全面测试，可以确保 RSS 阅读器的安全性、可靠性和易用性，提升 RSS 阅读器的用户体验。

表 6.2 用户注册测试项

| 测试用例            | 测试结果      | 记录结果 |
|-----------------|-----------|------|
| 填写新的用户名、密码和邮箱   | 注册成功      | 测试成功 |
| 填写重复的用户名、密码和邮箱  | 注册失败并显示错误 | 测试成功 |
| 填写不完整的用户名、密码和邮箱 | 注册失败并显示错误 | 测试成功 |
| 填写信息时，是否有格式限制   | 是         | 测试成功 |
| 注册后，页面是否正确跳转到登录 | 是         | 测试成功 |

### 6.3 主页分组和文章更新测试

该测试项的目的是验证在主页中添加分组和文章更新功能是否正常，需要测试以下方面：

表 6.3 主页分组和文章更新测试项

| 测试用例                     | 测试结果 | 记录结果 |
|--------------------------|------|------|
| 添加新的分组，将多个 RSS 源添加到不同分组中 | 添加成功 | 测试成功 |
| 确认添加的 RSS 源可以正确地显示在对应分组中 | 正确显示 | 测试成功 |
| 是否返回更新文章数，最新的文章排在前面      | 是    | 测试成功 |
| 每个分组的文章是否只包含该分组内的 RSS 源  | 是    | 测试成功 |

## 6.4 RSS 源管理功能测试

RSS 源管理功能测试是验证 RSS 源管理功能否正常工作，需要测试以下方面：

表 6.4 RSS 源管理功能测试项

| 测试用例                                    | 测试结果 | 记录结果 |
|-----------------------------------------|------|------|
| 添加新的 RSS 源，RSS 源内容正确显示在表格和分组中           | 正确显示 | 测试成功 |
| 修改已有的 RSS 源信息，更新后 RSS 源内容是否正确地显示在表格和分组中 | 正确显示 | 测试成功 |
| 删除已有的 RSS 源，该 RSS 源的内容是否不再显示            | 是    | 测试成功 |
| 查询已添加的 RSS 源，是否能正确地返回结果                 | 是    | 测试成功 |

## 6.5 文章列表和阅读功能测试

文章列表和阅读功能测试的目的是验证文章列表是否能正常显示，阅读时各种功能按钮能否正常工作。在进行文章列表和阅读功能测试时，需要测试以下方面：

表 6.5 文章列表和阅读功能测试项

| 测试用例                            | 测试结果 | 记录结果 |
|---------------------------------|------|------|
| 点击侧边栏的分组内的 RSS 源名称，检查是否正确显示文章列表 | 正确显示 | 测试成功 |
| 点击某篇文章的标题，是否能显示文章内容             | 是    | 测试成功 |
| 点击按钮标记文章为已读或未读，检查标记是否正确         | 标记正确 | 测试成功 |
| 点击按钮收藏文章或取消收藏文章是否正常             | 正常   | 测试成功 |
| 点击跳转原文按钮能否跳转到该文章的原文页面           | 正确跳转 | 测试成功 |

## 6.6 收藏历史功能测试

收藏历史功能测试目的是验证查询收藏历史的功能能否正常工作。在进行收藏历史功能测试时，需要测试以下方面：

表 6.6 收藏历史功能测试项

| 测试用例                   | 测试结果 | 记录结果 |
|------------------------|------|------|
| 点击查看收藏的所有文章，是否能正确地返回结果 | 是    | 测试成功 |
| 取消收藏一篇文章，是否成功取消收藏。     | 是    | 测试成功 |
| 点击某篇文章的标题，是否能显示文章内容    | 是    | 测试成功 |
| 点击跳转原文按钮能否跳转到该文章的原文页面  | 正确跳转 | 测试成功 |

## 6.7 历史阅读功能测试

历史阅读功能测试目的是验证查询阅读历史的功能能否正常工作。在进行历史阅读功能测试时，需要测试以下方面：

表 6.7 历史阅读功能测试项

| 测试用例                  | 测试结果 | 记录结果 |
|-----------------------|------|------|
| 点击查看已读的文章，是否能正确地返回结果  | 是    | 测试成功 |
| 将某篇已读文章标记为未读，是否成功标为未读 | 是    | 测试成功 |
| 点击某篇文章的标题，是否能显示文章内容   | 是    | 测试成功 |
| 点击跳转原文按钮能否跳转到该文章的原文页面 | 正确跳转 | 测试成功 |

## 结束语

在论文选题时，发现了这个令我感兴趣的课题，我希望可以实现一个 RSS 阅读器。RSS 阅读器对我来说是一个充满挑战但也非常有收获的经历。一开始，我对于完成这个项目所需要的知识和技能感到不确定和困惑。不过，通过持续地努力和愿意学习的态度，我成功地克服了困难并完成了我的目标。

在开发过程中，我遇到了各种技术难题，但是也依靠在线资源和自学来找到解决方案。在此过程中学习并理解了 SpringMVC 和 Java 的 Rome 库。也知道了各种实现 RSS 阅读器的技术栈和框架。

构建这个 RSS 阅读器的经历教会了我软件开发方面宝贵的经验，并提高了我在 Java Web 开发方面的技能。虽然还有很大的提升空间，但我为自己所取得的成就感到自豪，也相信这个项目为我未来职业生涯中可能面临的挑战做好了准备。我坚信，通过不断的努力，我能够获得更多的成功，充分发挥我的潜能。

## 致 谢

本篇论文的完成是一个自我学习、认知和检验的过程，从开题选题到开题报告再到最终完成，都是经历了一番探索和挑战。在此，我要感谢我的指导老师，他严谨的治学精神、深厚的理论水平以及认真负责的工作态度对我产生了巨大的影响和启发。他的指导不仅在整体思路给了我很多帮助，也在关键技术提供了有益的指导和建议。同时，我还要感谢所有的任课老师，是他们的课堂教学和真挚教诲让我在知识的海洋里茁壮成长，成为一名合格的大学毕业生。我也要感谢我的同学和室友，他们在学习和生活上一直给予我帮助和支持，为我提供了宝贵的人生经验和回忆。我还要感谢毕业设计小组的同学们，虽然我们的课题各不相同，但是在完成毕业设计的过程中，我们互相帮助、互相支持，共同进步。我也要感谢答辩组的老师们，他们在百忙之中抽出时间为我们的答辩操心，为我们提供了宝贵的意见和建议。

由于我的学术水平有限，所写论文难免存在不足之处。在此，我真诚地恳请各位老师批评指正，希望能够得到您们的宝贵意见和建议，以便我在今后的学习和工作中不断提高自己。最后，再次向我的导师和所有任课老师们表示崇高的敬意和衷心的感谢！

## 参 考 文 献

- [1] 朱凯定. 经典的 RSS 订阅[J]. 计算机与网络, 2022, 48(02):32-33.
- [2] 刘镇宇. 基于 RSS 的图书馆推送服务系统的研究[J]. 数码世界, 2019, 161(03):92.
- [3] 殷存举. 基于 ASP.NET MVC 技术的在线 RSS 阅读器的设计与实现[J]. 信息技术与信息化, 2019, 236(11):93-95.
- [4] 李晶晶, 林静. 基于 C#的 RSS 移动新闻阅读器系统研究[J]. 电子测试, 2019, 425(20):59-60+32.
- [5] 黄春晓. 基于 inoreader 的科技信息聚合平台[J]. 农业图书情报学刊, 2018, 30(10):78-82.
- [6] 向梦宇. 基于 RSS 的企业档案知识服务模式研究[J]. 机电兵船档案, 2022, (2):43-45.
- [7] 张云华. 基于 RSS 技术的数学信息共享平台[J]. 数字技术与应用, 2021, 39(06):115-117.
- [8] 李根, 张磊. RSS 聚合和推送技术在公安情报工作中的应用探索[J]. 电脑与信息技术, 2017, 25(02):27-31.
- [9] 刘军, 金淑娜. 基于 RSS 订阅中心的高校图书馆网络信息资源聚合研究[J]. 图书馆学研究, 2013(19):60-63.
- [10] 郭吉楠. javaEE 企业级开发[J]. 电子技术与软件工程, 2018, (15):26-26
- [11] 李泉, 任维政. 基于 SpringMVC 的多平台 J2EE 开发方式研究[J]. 吉林大学学报: 信息科学版, 2017, 35(5):569-575.
- [12] 葛萌, 黄素萍, 欧阳宏基. 基于 SpringMVC 框架的 Java Web 应用[J]. 计算机与现代化, 2018, 276(08):97-101.
- [13] 杨艳华, 袁鹏飞. Oracle 11g 数据库的管理与开发基础教程[M]. 北京:人民邮电出版社, 2021.
- [14] Patrick Gotthard. Rome[EB/OL]. (2017-07-22) [2023-04-06]. <https://rometools.github.io/rome>.
- [15] 刘嘉琦. 探究 oracle 数据库的备份与恢复[J]. 数字技术与应用, 2017, (02):246.