

苏州大学

本科毕业设计(论文)

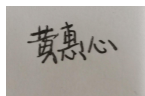
学院(部)	计算机科学与技术学院		
题 目	基于 Django 的博客系统开发		
年 级	2017 级	专业	计算机科学与技术
班 级	计算机三班	学号	1727405129
姓 名	黄惠心		
指导老师	韩冬	职称	副教授
论文提交日期	2021 年 5 月 11 日		

苏州大学

本科毕业设计（论文）独创性声明

本人郑重声明：所提交的本科毕业设计（论文）是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本设计（论文）不含其他个人或集体已经发表或撰写过的研究成果。对本文的研究作出重要贡献的个人和集体，均已在文中以明确方式标明。本人承担本声明的法律责任。

作者签名：



日 期：

2021/6/4

苏州大学

本科毕业设计（论文）使用授权声明

本人完全了解苏州大学关于收集、保存和使用本科毕业设计（论文）的规定，即：本科毕业设计（论文）的著作权以及文中研究成果的知识产权归属苏州大学。苏州大学有权向国家有关部门或第三方机构送交毕业设计（论文）的复印件和电子文档，允许毕业设计（论文）被查阅和借阅，可以采用影印、缩印或其他复制手段保存和汇编毕业设计（论文），可以将毕业设计（论文）的全部或部分内容编入有关数据库进行检索。

涉密设计（论文）☐

本设计（论文）属 _____ 在 _____ 年 _____ 月解密后适用本规定。

非涉密设计（论文）☐

论文作者签名：  日期： 2021/6/4

导师签名：  日期： 2021/6/4

摘要

随着现代科技的发展，互联网的使用逐渐普及，互联网技术日新月异，因此各类网络交流平台也在不断改进。博客系统通过多样化的元素为各种各样的人群提供一个可以共同使用的平台，提供了快捷便利的交流通道，也为每个人提供了一个可以展示自我的机会，这顺应了互联网的发展趋势，对知识的扩散和传播具有重大意义。在此背景下，基于博客方便使用和轻松社交的特点，又基于 Django 框架的易于开发和维护的特点，设计和实现了本次基于 Django 的博客系统。

根据软件的开发流程，本文首先对系统进行了可行性分析和系统目标分析，然后分别对系统的功能需求、数据需求、角色需求和非功能性需求进行分析。接着对系统进行了总体设计，分别详细分析了每个模块的具体设计，绘制系统的流程图和结构图帮助完成系统设计。借助开发工具实现了包括用户注册和登录、发表博文、评论、文章审查等博客系统基本功能。最后，借助软件测试方法本文对该系统进行了测试。

综上所述，本项目已基本完成一个博客系统的基础功能开发。借助 Django 框架进行系统的开发，能够提高开发效率，方便后期维护。前端使用 Bootstrap 框架，界面简洁美观、用户操作简单。写博客能够督促人进步，促进知识的传播，记录生活的点滴，和志同道合的朋友交流，帮助别人，因此研究和设计博客系统具有一定的使用价值和意义。

关键词：Django 框架；博客系统；知识分享

Abstract

With the development and innovation of technology, the Internet is becoming more and more popular and its technology is changing day by day, so all kinds of online communication platforms are constantly being improved. The blogging system provides a common platform for all kinds of people to use through a variety of elements, providing the quickest and most convenient communication channel and an opportunity for everyone to showcase themselves. It is of great significance for the spread and dissemination of knowledge. In this context, based on the simple expression and easy social features of blogs, and also based on the easy development and maintenance features of the django framework, the current django-based blogging system was designed and implemented.

According to the software development process, this paper first analyses the feasibility and system objectives of the system, then analyses the functional requirements, data requirements, role requirements and non-functional requirements of the system respectively, and carries out the overall design of the system, analyses the specific design of each module separately in detail, draws the flowchart and structure diagram of the system to help complete the system design. With the help of development tools, the basic functions of the blog system, including user registration and login, posting, commenting and article review, were implemented. Finally, the system has been tested with the help of software testing methods.

In summary, this project has basically completed the development of the basic functions of a blog system. The Django framework is used to develop the system, which can improve the development efficiency and facilitate maintenance later. The front-end uses Bootstrap framework, with a simple and beautiful interface and simple user operation. Blogging can help people to make progress, promote the spread of knowledge, record their lives, communicate with like-minded friends and help others, so the research and design of the blog system has a certain value and significance.

Keywords: Django framework; blogging system; knowledge sharing

目 录

1 绪论.....	1
1.1 研究背景.....	1
1.2 研究意义.....	1
1.3 国内外研究现状.....	2
1.4 主要研究内容.....	3
2 关键技术分析.....	4
2.1 python 语言.....	4
2.2 Django 框架.....	4
2.2.1 Django.....	4
2.2.2 Django ORM.....	4
2.2.3 Django SQLite3.....	5
2.3 MVC 设计模式.....	5
2.4 Bootstrap.....	5
3 系统需求分析.....	6
3.1 可行性分析.....	6
3.2 系统目标.....	6
3.3 功能性需求分析.....	7
3.4 数据需求分析.....	9
3.4.1 数据流图.....	9
3.4.2 数据字典.....	9
3.5 角色需求分析.....	11
3.6 非功能性需求分析.....	13
4 系统设计.....	15

4.1 系统设计原则.....	15
4.2 系统开发架构设计.....	15
4.3 系统功能结构设计.....	15
4.4 系统流程图设计.....	16
4.5 系统模块设计.....	17
4.6 系统数据库设计.....	19
5 系统实现.....	22
5.1 系统前台功能模块的实现.....	22
5.1.1 用户管理模块.....	22
5.1.2 文章管理模块.....	27
5.1.3 互动管理模块.....	31
5.2 系统后台管理模块的实现.....	32
6 系统测试.....	34
6.1 开发环境和系统部署.....	34
6.2 测试.....	35
6.2.1 系统测试方法.....	35
6.2.2 功能测试.....	35
6.2.3 兼容性测试.....	36
7 总结与展望.....	37
7.1 全文总结.....	37
7.2 展望.....	37
参考文献.....	38
致谢.....	40

1 绪论

1.1 研究背景

随着互联网技术快速发展，我们的生活工作学习都产生了巨大的变化，传统的报纸、书籍等媒介已经越来越无法满足人们日新月异的交流需要。于是，在网络上进行交流逐渐成了人们更加偏爱的方式，网络博客就是其中一种交流手段。因此，博客是时代不断发展的产物，是随着人们的需求出现的。

博客，就是利用互联网，人们可以在上面分享自己的心得、趣事等，也可以与其他人进行实时互动。从具体的定义来说，博客就是一种软件，人们在这个软件上能够发表、传播个人的文章^[1]。作为近些年的一种新的内容丰富且个性化的网络交流平台，博客受到了越来越多的人的欢迎和喜爱，改变了我们的一些生活方式和学习方式。博主通过博客来分享自己的心得、创作，其他人可以通过评论交流、发表自己的看法。同时，博客也可以作为学习知识的平台，大家可以在博客上分享知识、经验。博客具有实时化、个性化的特点，同时它又简单易用，因此博客受到了广泛欢迎。

如今比较知名的博客网站有 CSDN：CSDN 博客是专为 IT 从业者、IT 学习者打造的供人们交流互联网技术、学习专业知识的博客平台，使开发者能够通过这个平台分享和学习互联网知识。LOFTER：LOFTER 是网易推出的一款轻博客，具有网页版和 APP 两种形式，口号是“专注兴趣，分享创作”。主要吸引年轻人分享自己的原创作品，人们可以在这里根据自己的兴趣找到志同道合的朋友。新浪微博：新浪微博是一种微型博客，它只能发表 140 字左右的内容。微博的特点是使用门槛低且能随时随地实现快速传播，这种方便快捷的形式受到了越来越多的年轻人的喜欢。

随着科学技术的发展，博客技术也随之发展，最开始的博客使用 C 或 C++ 编写，后来 PHP 开始流行，最近几年 Python 也开始被广泛用来实现博客系统的搭建^[2]。

1.2 研究意义

每个人对于博客都有不同的个性化需求，本项目除了拥有博客系统的基本功能外，还添加了一些功能设计，使系统更加人性化，能够提高用户的使用体验。

在本文的博客系统中，为了让用户更方便的看到最热门的内容，根据浏览量对文章进行最热文章推荐，可以使用户最快速的看到热门的文章。考虑到博客的社交功能，本系统为博客添加了标签功能和分类功能，用户可以通过不同的标签和分类找到自己喜欢的文章，聚集拥有同样兴趣的群体，在标签和分类下探讨共同的话题。由于使用博客的人群多样，有时可能会发布一些违规内容，为了尽力避免这种情况，本系统为文章发表增添了文章审查功能。当文章内容中包括一些敏感内容时将会禁止发表并对用户进行提醒。另外，为了提高用户的使用体验，系统为用户增添了音乐播放等人性化小功能设计。

本次设计利用 Django 框架完成了一个博客系统的创建，体现了 Django 在开发时的特点，证明了 Django 和 Python 的技术已经发展成熟，通过 Django 框架开发属于自己的博客网站已经可行。同时能够证明利用 Django 开发网站的优越性，可以使复杂的网页开发简单化，能够提高开发者的效率，减少工作量，缩短开发周期。

1.3 国内外研究现状

博客的起源是在 1997 年的美国，一位人工智能的专家提出了 Weblog 这个词，它的含义是网络日志，后来演化成了 Blog 也就是博客^[3]。由于博客是人们进行知识交流和分享的重要工具，因此国内外对于博客的技术和内容研究一直很热门。

由于国内博客系统的起步比较晚，因此国内对于博客系统的研究也比较晚。国内的博客开始流行是在 2005 年以后，网易、搜狐等网站开始进行博客系统的设计和开发，但是发展的也并不顺利，到如今轻博客广泛流行，最有代表性的就是新浪微博。国内对于博客系统的研究可以分成几个方向。

（1）对于博客本身的研究。博客的类型，博客的使用等。

（2）对于博客用户的研究。例如：刘立行、张诗的《视频博客阅听众使用行为研究之初探》^[4]主要探讨了视频博客的用户特征、使用动机和使用情况。雷霄主要研究了网络趣缘群体对网易 LOFTER 的使用感受^[5]。

（3）博客所涉及的法律问题。例如季渝所写的《论微博作品的著作权保护》^[6]就是对微博作品的著作权保护问题的探讨。

（4）博客的影响力研究。例如《学术博客影响力评价研究——以科学网博客为

例》^[7]（王琛）、《微博客对新闻传播的影响》^[8]（佟文娟、宁健铭）等。

（5）对博客的开发进行的研究，包括基于各种开发框架和技术对博客系统进行构建。例如：《基于 Django 的个人博客系统设计开发》^[9]（常佳宁、李阳齐）主要采用 Django 框架和 MVC 设计模式实现博客。涂远杰、郑剑主要是利用 Flask 框架，同时使用了爬虫和 SQLite 数据库，前端使用 HTML、JavaScript 开发^[10]。另外李嘉明基于 Node.js 实现的多人博客系统^[11]，和鞠宏军、林涛基于 Spring Boot 和 Hibernate 框架设计和实现博客系统^[12]等等。

1.4 主要研究内容

本文的主要内容是对基于 Django 的博客系统进行分析和设计。通过软件工程的开发流程，借助 Pycharm 工具使用 Python 高级语言编写代码，使用了目前比较流行的 Django 框架，具体研究内容如下：

- （1）对研究背景和研究意义进行阐述，然后介绍国内外对博客系统的相关研究。
- （2）对本次开发的博客系统使用的关键性技术进行学习和研究，为后面博客系统的设计和开发做铺垫。
- （3）对博客系统进行可行性分析和需求分析，阐述了系统的目标。
- （4）介绍博客系统的系统设计，包括功能设计、流程图设计、模块设计、数据库设计等内容。
- （5）实现博客系统，利用文字和相关图表来说明系统的功能和实现过程。
- （6）对实现后的博客系统进行测试，通过功能测试和系统兼容性测试来测试本博客系统是否达标，利用测试用例具体说明测试过程。最终结论是本博客系统满足了用户的基本需求，达到了设计的要求。
- （7）总结了现阶段的开发成果，找出不足之处和可以继续改进的地方，展望系统未来的发展方向。

2 关键技术分析

2.1 Python 语言

Python 是由吉多·范罗苏姆开发的一种面向对象的、交互式的程序设计语言，因其语法简单、可读性好的特点很受初学者青睐^[13]。Python 是解释型的语言，可以跨平台使用，能够在多种操作系统中运行。提供所有主流的数据库接口，非常适合大型程序开发。Python 拥有丰富和强大的框架和类库，同时，它也是一种功能强大的编程语言，如今 Python 已经广泛应用于各种 Web 编程中。它在软件分析、设计、编码、测试、维护等环节都起到了非常大的作用，极大地提高了开发效率^[14]。

2.2 Django 框架

2.2.1 Django

Python 中的常见 Web 编程框架有 Django、Tomado、Nask 等，目前使用最多的框架就是 Django。Django 是用 Python 写的、开放源码的应用框架，能够简单快速的开发一个网站。采用 MVC 模式，重视代码的复用，可扩展性强。核心组件有：模板语言、ORM、Admin 管理、Url 映射、表单管理。Django 能够使开发复杂的网站变得简单便捷^[15]。

Django 开发的优势在于（1）功能完备，包括 ORM、视图模板、路由映射、表单等。（2）通用，可以构建各种类型的网站，可以和其他客户端框架一起工作。（3）安全，能够避免许多错误，保护网站。（4）可移植，不受操作系统限制。（5）自动管理后台^[16]。

2.2.2 Django ORM

ORM（Object Relational Mapping）的意思是对象关系映射，主要用来描述数据模型类和数据库之间的对应关系，就是让一个模型类和一个数据库表一一对应。ORM 是数据库层和业务逻辑层之间的桥梁^[17]。Django 通过类描述数据库的字段、表间关系等内容，通过命令把类描述的内容传入数据库。

Django ORM 提供了通过类对象操作数据库的方式，能够将数据库语句转换成比较固定的 Django 语法结构，能够提高开发人员的开发效率，但是由于最终还是需要将这些语句转化成 SQL 语句，会牺牲一些执行效率^[18]。

2.2.3 Django SQLite3

本次开发的是博客系统，选择 Django 的内置数据库 SQLite3。SQLite3 是一个轻量级的数据库，体积小，速度快，方便移动，支持不同操作系统，占用少量资源，方便开发调试。

2.3 MVC 设计模式

MVC 模式是比较流行的一种设计模式，Django 就是按照 MVC 模式开发的框架^[19]。MVC 模式就是把 Web 分成三层：模型 Model、视图 View、控制器 Controller。

模型是数据存储层，提供数据存储接口，模型从数据库中取数据时，不需要知道数据库取数据的方式。Models.py 创建数据库模型，处理关于存取方式、有效性、数据关系和做与数据相关的事务。

视图是界面，是模型的表现层，用于处理显示什么和怎么显示。实现视图用到的技术有 HTML、CSS、JS 等前端技术。

控制器负责业务逻辑，通过逻辑结构决定从数据库中取什么数据，传递什么信息给视图。Urls.py 定义 Url 路由表，通过用户请求调用 Views.py 中的函数与数据模型和视图交互，响应用户请求。

2.4 Bootstrap

Bootstrap，来自 Twitter，是目前最受欢迎的前端框架。Bootstrap 基于 HTML、CSS、JavaScript，由于它简洁灵活的特点使得 Web 开发更加快捷。所有的主流浏览器都支持 Bootstrap。Bootstrap 包含了十几个可重用的组件，用于创建图像、下拉菜单、导航、警告框、弹出框等等。

3 系统需求分析

前两章对分别对研究背景和相关技术进行了介绍，本章主要对系统进行需求分析。主要包括可行性分析、系统目标、功能性需求分析、数据需求分析、角色需求分析和非功能性需求分析。

3.1 可行性分析

一个项目开发前进行合理的可行性分析是必不可少的。一个系统需要确定用户定位，判断开发项目是否满足用户需求，也要分析项目开发是否可行。本系统的具体可行性分析如下：

从使用可行性分析，本博客系统用户操作简单，维护成本低，方便用户管理自己的个人博客。通过将流行的 Django 开发技术与博客系统的结合，能够推进个人博客系统的进一步发展。

从经济可行性分析，本博客系统对开发的设备和环境要求不高，只需要一台个人计算机就可以进行开发，开发后的维护也很方便。和传统的大型博客相比，需要花费的成本非常少，因此基于 Django 开发博客系统在经济上是可行的。

从技术可行性分析，本博客系统使用的是 Python 中的稳定框架 Django 进行开发，Django 技术目前已经十分成熟，能够提高网站的开发效率，并且 Python 语言具有初学者容易上手的优点。在选用数据库时，选用 Django 内置的数据库 SQLite3，解决了数据存储的问题。页面文件的编写和一些交互的实现也涉及 HTML、JavaScript、CSS。这些技术都是当前主流的开发技术，都是效率高、简洁、功能完备的技术，因此利用这些技术进行博客系统开发是可行的。由于这些技术已经比较完备，且使用者众多，因此即使在开发中开发者遇到问题，也可以通过互联网搜索来解决。本系统选择 Pycharm 为开发软件，它具有完整的编码、调试、发布等一系列开发功能，能够大大提高开发者的开发效率。由此可见此博客系统的开发在技术上是可行的。

从社会可行性分析，由于人们对此类博客系统有需要，因此本博客系统具有社会可行性。

3.2 系统目标

系统总体目标是设计并开发一款结构合理，技术先进，功能完整，方便维护，用户体验好的博客系统网站。使用过程中不会轻易随着时代的发展而被淘汰，用户界面设计简洁美观，具有一定的风格且功能简单不复杂，方便用户学习和使用。希望能够为用户打造一个分享个人心得，能够实时交流，能够学习知识，能够网上交友的开放和谐的社交平台。具体目标如下：

- （1）支持多用户访问，每个人都可以注册获得自己的个人主页，登录后都能够获得编写并发表文章的功能，并且可以为自己的文章选择个性化的分组和标签。
- （2）数据库的选择方便操作，数据安全。
- （3）定义明确，方便扩展，优化代码，提高代码可读性，降低冗余。
- （4）系统的前台模块条理清晰，系统构架完整，内容功能完善。功能设计合理，平衡分享和社交需求。

3.3 功能性需求分析

从系统的可行性和系统目标分析后就要进行系统的功能需求分析。

从博客系统的功能需求的角度分析，本博客系统主要由以下几个基本功能模块组成：文章管理功能、用户管理功能、互动管理功能、系统管理功能。管理员可以在后台对博文信息进行增加、修改和删除，方便维护网站也方便监督网站用户行为的文明合法。另外，为了博客系统的美观，还需要使用 HTML、CSS 和 JavaScript 对系统的前端界面进行设计。具体的需求分析如下：

文章管理功能

- （1）最新文章：按照文章最后发布的顺序显示文章标题，并链接到文章详情。
- （2）最热文章：将会按照浏览量对文章热度进行排序，并在栏目中显示当前最热的五篇文章，点击全部最热按钮即可看到全部文章热度排序。
- （3）文章分类：按照文章内容将文章分成不同类别，发布文章时选择文章类别，一篇文章只能对应一个分类。分类栏目显示所有文章类别，并链接到属于此类的所有作者按时间排序的文章列表。用户可以通过新增分类按钮添加新的分类。

（4）文章标签：设置多个不同的文章标签，发布文章时可以为文章贴标签，一篇文章可以贴多个标签。标签栏显示所有标签，并链接到与此标签相关的按时间排序的所有文章列表。用户可以通过新增标签按钮添加新的标签。

（5）时间归档：按照文章的年月份对所有文章归档，并可以显示此月份下发表的所有文章。

（6）文章发布：用户点击新增文章按钮编写文章内容，填写文章标题、摘要，选择文章分类、标签。文章内容用户可以上传图片、表格等。本系统的特点在于能够对用户即将发布的文章审查，管理员通过在后台设定敏感词，审查用户中是否出现违规内容，对违规文章不允发布，并提醒用户文章包含违规内容。

（7）文章删除：普通用户可以删除自己发布的文章，不可以删除别人的文章。

（8）编辑文章：用户可以点击文章题目下方的编辑文章按钮进入编辑文章界面，对文章的标题和内容进行修改，每个用户只能编辑自己发表的文章，非本用户发表的文章将没有修改文章按钮。

（9）我的文章：显示所有文章作者为此登录账号的文章列表，没有登录的用户将无法看到此界面。

（10）文章搜索：通过搜索框使用关键词搜索文章内容包括关键词的文章并显示文章详情。

（11）文章信息：包括文章发布时间、文章作者、文章阅读量等文章其他信息。其中文章阅读量每次打开文章详情阅读量都会加一。管理员在后台可以对分类、标签、文章进行增加、删除和修改。

用户管理功能

用户包括游客、普通用户和管理员。游客只能浏览文章，不能发表文章或对文章发表评论。登录后的用户可以发表文章和评论，并且拥有显示我的文章列表功能。用户信息包括：账号、密码、姓名、邮箱、电话、头像。每个用户拥有一个唯一的 ID，此后的身份识别都是通过这个唯一的 ID 实现的。注册和登录的界面主要以简洁易操作为主，用户从前台提交数据后，后台存储在数据库中，登录时比对数据库中的信息，账号密码一致时可以登录。每个登录用户拥有自己的个人资料卡，将会展示用户的姓名、邮箱、联系方式等个人信息，用户可以修改自己的姓名、电话、邮箱，没有登录

的用户将不会拥有个人资料卡。管理员可以在后台进行增加用户、删除用户、修改用户信息，并为不同的用户分配相应的权限。

互动功能

已经登录的用户可以评论文章，这一模块体现了博客系统的社交功能。用户可以删除自己发布的评论，后台可以管理所有评论信息。

系统管理功能

主要是管理员在系统后台对博客系统进行统一管理，主要包括文章、评论、敏感词、分类、标签相关信息的增删改操作。

3.4 数据需求分析

3.4.1 数据流图

数据流图能够通过数据转换的逻辑关系展示系统内部相关模块的逻辑关系，数据流图经常被用来进行数据需求分析。

根据上述需求分析，画出数据流图如图 3-1 所示。

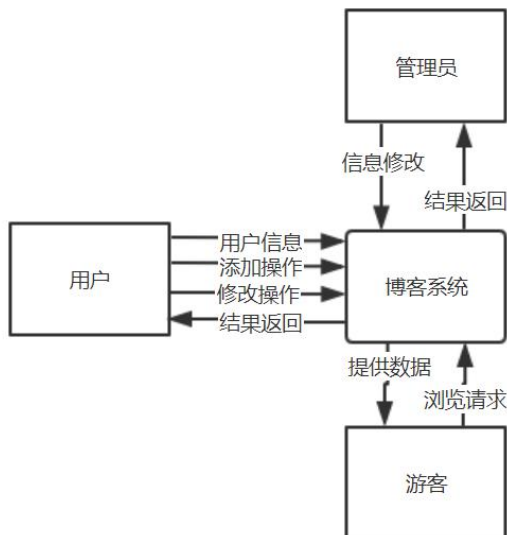


图 3-1 博客系统整体数据流图

3.4.2 数据字典

数据字典用来对数据流图的成分详细说明，是对数据流图的补充，帮助开发者理解数据的具体意义。以下是根据博客系统需求分析列出的数据字典。

用户信息表主要描述网站中用户的基本信息，如表 3-1 所示。

表 3-1 用户信息表

名字	用户信息表
内容	ID、账号、密码、邮箱、昵称、权限、电话、头像
位置	后台管理模块、前台用户资料

分类表主要描述博客的分类信息，如表 3-2 所示。

表 3-2 分类表

名字	分类表
内容	ID、名称、备注
位置	后台管理模块、前台管理分类模块

标签表主要描述博客的标签信息，如表 3-3 所示。

表 3-3 标签表

名字	标签表
内容	ID、名称、备注
位置	后台管理模块、前台标签管理模块

文章表主要用于描述博客的文章相关信息，如表 3-4 所示。

表 3-4 文章表

名字	文章表
内容	ID、题目、内容、创建时间、修改时间、摘要、浏览次数、作者 ID、分类 ID
位置	后台管理模块、前台文章管理模块

敏感词表主要用于表示敏感词的相关信息，如表 3-5 所示。

表 3-5 敏感词表

名字	敏感词表
内容	ID、名称
位置	后台管理模块、前台文章管理模块

评论表主要用于描述评论的相关信息，如表 3-6 所示。

表 3-6 评论表

名字	评论表
内容	ID、邮箱、评论内容、评论时间、文章 ID、评论者
位置	后台管理模块、评论管理模块

3.5 角色需求分析

从用户角色的角度分析系统的具体功能，将用户划分成游客、普通用户、管理员三种不同的类型。

游客是指尚未在当前网站注册的使用者或者已经注册过但尚未登录的用户，也就是当前网站的匿名访问者。由于他们没有自己的身份标识，因此普通游客使用网站只拥有浏览博客、搜索博客功能，由于系统设置的权限，游客不能发表文章也不能对文章进行匿名评论。游客的用例图如图 3-2 所示。

用例分析说明：

- （1）浏览博文，游客可以查看当前系统中所有发表的文章。
- （2）搜索博文，可以对数据库中的所有博文进行检索并展示。

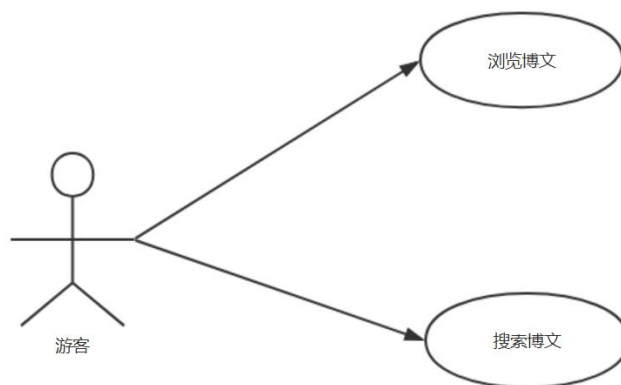


图 3-2 游客用例图

用户登录后就属于普通用户，除了拥有游客的权限之外，登录后的用户还拥有发表博文、编辑博文、发表评论、个人资料等专属功能。用例图如图 3-3 所示。

用例分析说明：

- （1）博文管理，普通用户拥有权限对系统中文章进行的操作，主要包括浏览博

文、搜索博文、发表博文、修改博文、删除博文。

（2）浏览博文，普通用户拥有和游客一样的浏览博文权限。

（3）搜索博文，可以对当前系统中的博文进行文章检索。

（4）发表博文，用户可以发表博客，并且根据自己的喜好设置博客的标题、摘要、博客内容。为了方便对博客的划分，也为了方便的查找相关博客，用户在发表博文时可以对博客选择相应的分类和标签。

（5）修改、删除博文，用户可以编辑修改和删除自己发表的博文。

（6）个人资料，登录后的用户可以查看自己的个人资料卡，并有权修改自己的基本信息。

（7）评论，登录后的用户可以发表文章评论，可以删除自己发布的评论。

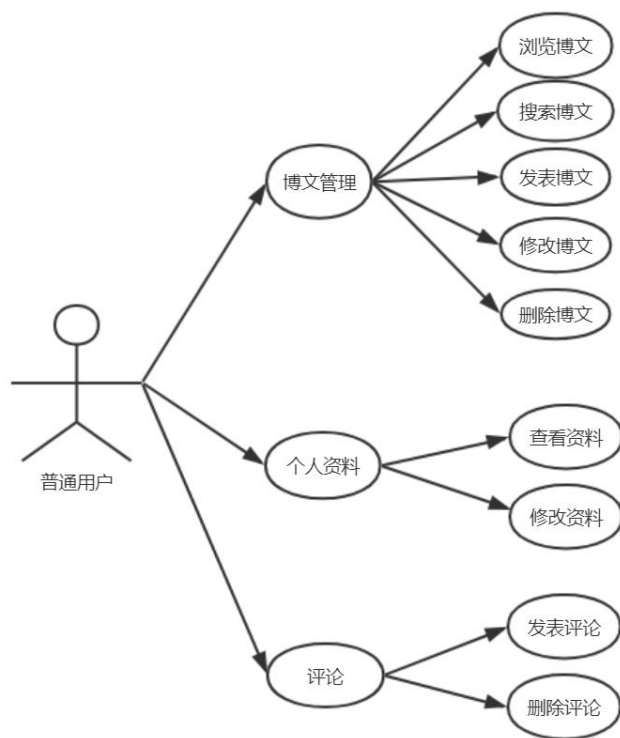


图 3-3 普通用户用例图

管理员是指博客系统的管理员，它不仅拥有普通用户的权利，还拥有管理文章、标签、分类、敏感词、用户、评论的超级功能。管理员的用例图如 3-4 所示。

用例分析说明：

（1）用户管理，管理系统中的所有用户，包括添加新用户、用户信息修改、用户权限修改、删除用户操作。

（2）文章管理，对系统中的文章进行管理，包括新增、修改、删除文章等操作。

（3）分类、标签管理，管理博客系统中的分类和标签，进行分类和标签的增删改操作。

（4）敏感词管理，为了方便文章的审查，管理员可以为文章设定一些敏感词，并通过后台对敏感词进行添加、修改、删除操作。

（5）评论管理，管理员可以对用户发表的评论增删改。

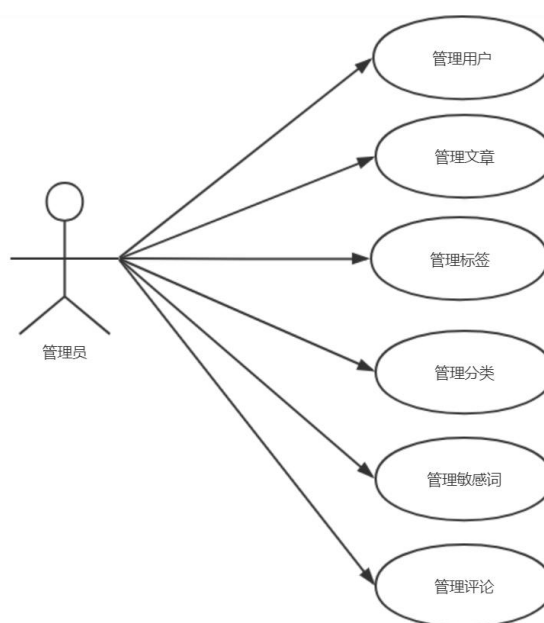


图 3-4 管理员用例图

3.6 非功能性需求分析

（1）界面规范：数据表单简洁规范，操作简单易懂。界面以蓝色浅色系为主。主界面显示博客条数默认 5 条。具有音乐播放器、鼠标点击特效等提高用户体验感。

（2）浏览器需求：Google Chrome 浏览器、Microsoft edge 浏览器、IE 浏览器等主流浏览器均兼容。

（3）后期维护需求分析：由于在系统开发过程中，不可能做到万无一失，后期使用时可能会出现各种问题，因此开发完成并不代表结束，还需要很多后期的维护。为了满足用户可能产生的各种新的需要，应对使用时遇到的问题，需要我们的程序有较强的可扩展性，并经常对系统进行测试，保证后期遇到的问题及需要添加的业务都能够通过维护来解决。

（4）安全性：对用户输入的表单数据格式、数据正确性进行审查，对错误操作给出提示。对发表博客的内容进行审查，包含违规词的博客不允许发表。后台记录管理员对数据的变更操作的历史记录，方便对产生的问题原因进行追溯，防止因管理员误操作而造成数据丢失或异常。

4 系统设计

4.1 系统设计原则

经过需求分析后，就要对系统进一步规划，通过软件工程开发流程对系统进行整体和模块设计。系统设计原则主要包括完整性、规范化、易用性以及可扩展性原则，只有当满足这四种原则时，系统才能安全、易用、稳定^[20]。

4.2 系统开发架构设计

从系统架构上，系统主要被分为 Web 浏览器层、表示层、业务逻辑层、数据库层。用户通过浏览器来进行访问和操作。系统表示层主要利用 Bootstrap、HTML、JavaScript、CSS 实现，主要用于显示用户界面以及为用户提供交互接口。业务逻辑层方面通过 Django 框架用于与前后端数据交互，数据层采用 Sqlite3 数据库。具体系统架构框架如图 4-1 所示。

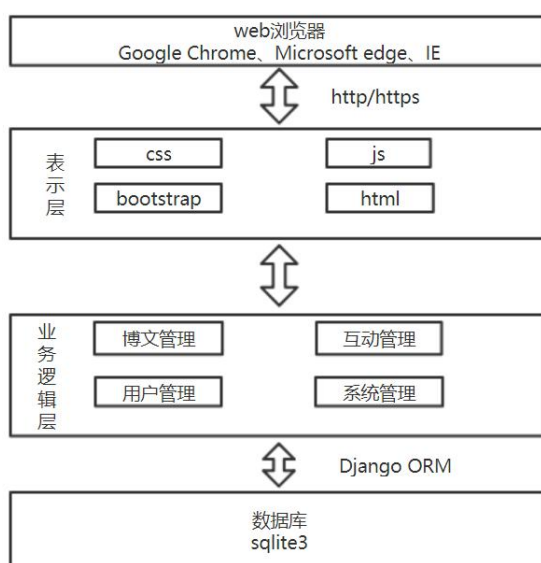


图 4-1 系统架构图

4.3 系统功能结构设计

基于 Django 的博客系统开发，除了发布博客、浏览博客等最基本的功能外，还添加了互动功能、标签功能等，通过前一章的需求分析，将博客系统分为博文管理、

用户管理、互动管理和系统管理四个模块。如图 4-2 所示。

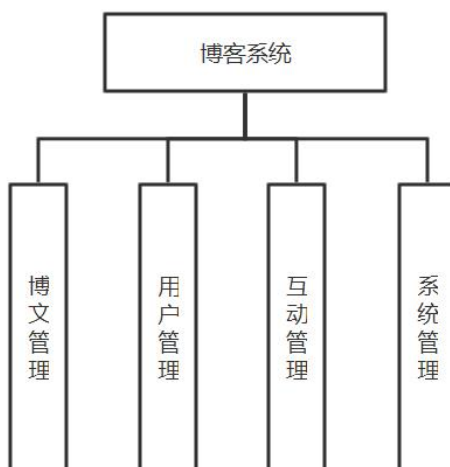


图 4-2 系统总体功能结构图

4.4 系统流程图设计

用户进入网站首先默认是游客身份，游客可以注册为普通用户。如果已经有注册帐户就可以直接登录。登录后，可以根据是否为管理员获得不同的权限。普通用户只能操作自己的博客，系统管理员可以在后台对所有系统数据执行管理操作。具体的流程图如图 4-3 所示。

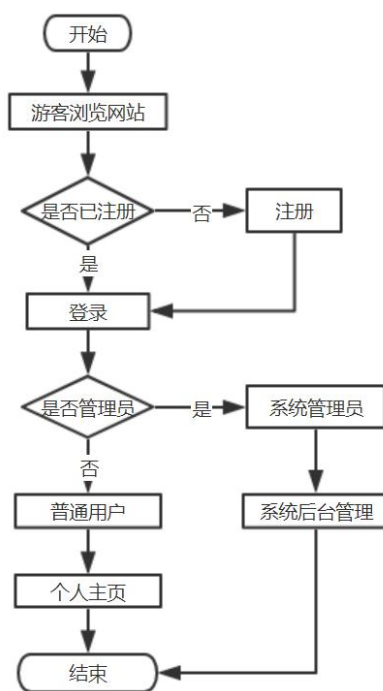


图 4-3 系统流程图

4.5 系统模块设计

本章对博客系统的四大模块进行详细分析设计。

（1）博文管理

博文管理主要指的是对系统内博客的一系列操作。不同用户拥有不同博文管理权限，普通用户只能管理自己的博客，管理员拥有所有权限。博文管理的功能结构图如图 4-4 所示。这里以发布博文为例给出流程图，如图 4-5 所示。

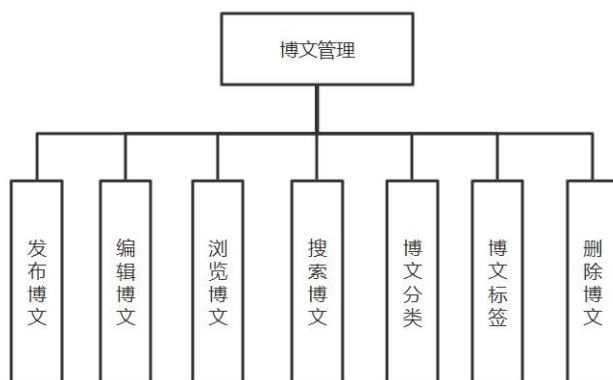


图 4-4 博文管理功能结构图

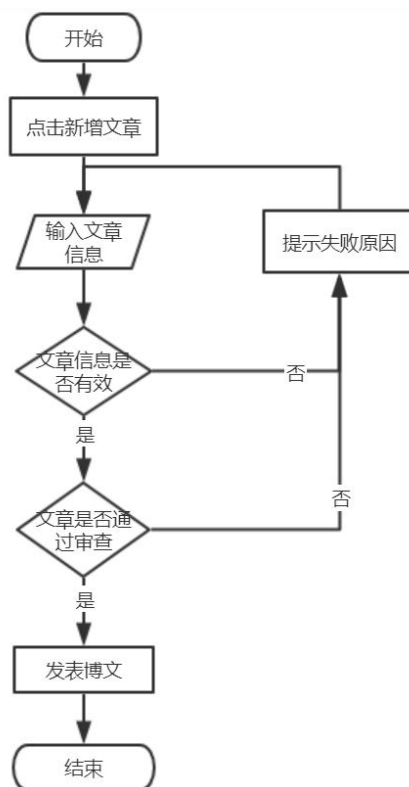


图 4-5 发表博文流程图

（2）用户管理

用户管理主要包括个人资料管理、用户账号管理、用户权限管理、注册登录管理。注册登录后，个人资料将显示用户的基本信息。用户 ID 是用来唯一区分不同用户的标准，并根据用户的不同类型给与用户相应权限。游客不能进行的操作，将在主页上不予以显示。用户信息可以通过后台修改。用户管理功能结构图如图 4-6 所示。

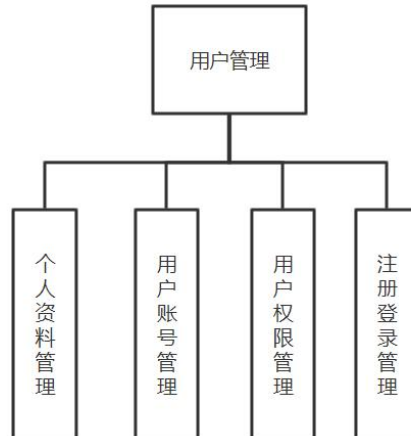


图 4-6 用户管理功能结构图

（3）互动管理

主要是评论管理，所有用户都可以查看已发布的评论。只有登录的用户才能发表评论。评论显示评论者的姓名和发表时间。用户可以删除自己的评论，管理员可以在后台更改评论信息。如图 4-7 所示。

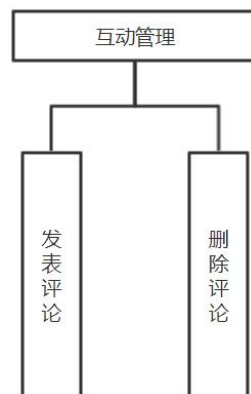


图 4-7 互动管理功能结构图

（4）系统管理

系统管理主要是指系统管理员进入后台才能进行的操作模块，具体包括用户信息管理、文章信息管理、分类信息管理、标签信息管理、敏感词管理、评论管理等操作。

系统管理功能结构图如图 4-8 所示。

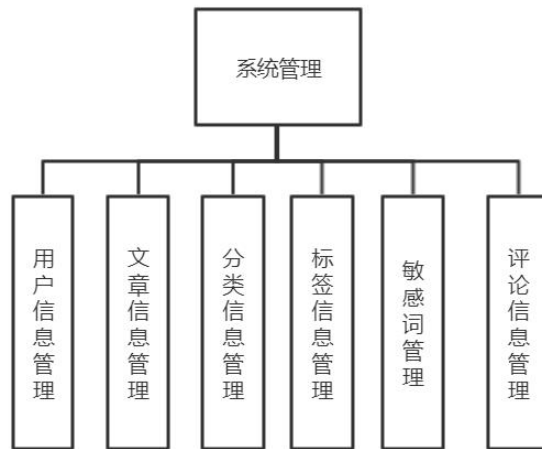


图 4-8 系统管理功能结构图

4.6 系统数据库设计

在这一节给出系统数据库相关表的设计，相关表设计如下。

（1）blog_loguser 用户表

blog_loguser 主要存储用户的个人信息，主要由用户账号、昵称、密码、邮箱等字段组成。如下表所示：

表 4-1 blog_loguser 用户表结构

字段名	数据类型	长度	是否主键	描述
id	integer		是	用户 ID
password	varchar	128	否	密码
last_login	datetime		否	上次登录时间
is_superuser	bool		否	是否超级用户
first_name	varchar	30	否	名
last_name	varchar	150	否	姓
email	varchar	254	否	邮箱
is_staff	bool		否	是否管理员
is_active	bool		否	是否可用
date_joined	datetime		否	注册日期

字段名	数据类型	长度	是否主键	描述
nickname	varchar	32	否	昵称
telephone	varchar	11	否	电话
head_img	varchar	100	否	头像
username	varchar	150	否	用户名

（2）blog_blog 文章表

blog_blog 用于存储有关文章的信息，主要包括标题、内容、摘要、作者、分类等字段。如下表所示：

表 4-2 blog_blog 文章表结构

字段名	数据类型	长度	是否主键	描述
id	integer		是	文章 ID
title	varchar	70	否	标题
body	text		否	内容
create_time	datetime		否	创建时间
modified_time	datetime		否	修改时间
excerpt	varchar	200	否	摘要
views	integer		否	浏览量
author_id	integer		否	作者 ID
category_id	integer		否	分类 ID

（3）blog_category 分类表

blog_category 用于存储分类信息，主要包括分类 ID、分类名、备注等字段。如下表所示：

表 4-3 blog_category 分类表结构

字段名	数据类型	长度	是否主键	描述
id	integer		是	分类 ID
name	varchar	32	否	分类名
des	varchar	100	否	备注

（4）blog_tag 标签表

blog_tag 是用来存储标签信息的，主要由标签 ID、标签名、备注等字段组成表结构。如下表所示：

表 4-4 blog_tag 标签表结构

字段名	数据类型	长度	是否主键	描述
id	integer		是	标签 ID
name	varchar	32	否	标签名
des	varchar	100	否	备注

（5）blog_black 敏感词表

blog_black 是用来存储敏感词信息的，主要由敏感词 ID、敏感词等字段组成表结构。如下表所示：

表 4-5 blog_black 敏感词表结构

字段名	数据类型	长度	是否主键	描述
id	integer		是	敏感词 ID
name	varchar	32	否	敏感词名

（6）comments_comment 评论表

comments_comment 是用来存储评论信息的，主要由评论内容、评论时间、邮箱、评论者等字段组成表结构。如下表所示：

表 4-6 comments_comment 评论表结构

字段名	数据类型	长度	是否主键	描述
id	integer		是	评论 ID
name	varchar	32	否	评论者姓名
email	varchar	60	否	邮箱
text	text		否	内容
created_time	datetime		否	评论时间
blog_id	integer		否	文章 ID

5 系统实现

在前一章本文已经完成了系统设计的内容，本章主要从前台功能模块和后台管理模块介绍系统功能的实现。

5.1 系统前台功能模块的实现

一个系统想要成功，一个好的界面是必不可少的。用户的功能操作都是在前端实现的，网站的前台设计会给使用者留下对系统的第一印象，也会对用户的使用体验产生影响。

5.1.1 用户管理模块

（1）注册

注册模块是系统必须的模块，用户在主界面的导航栏中单击注册按钮跳转到注册页。注册页面具体如图 5-1 所示

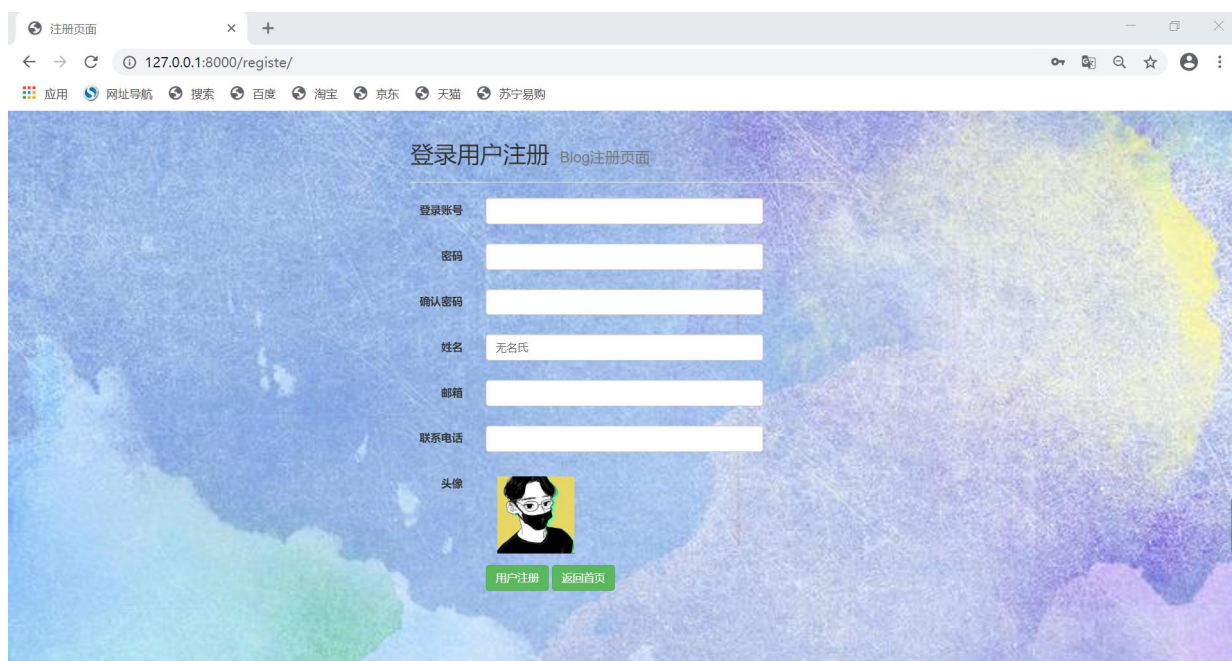


图 5-1 注册界面图

此时，用户输入信息后注册，我们为用户信息设定了一些约束。

- 1) 如果登录账号已经被注册过，从数据表查询有同名记录，则在注册时将会不成功，并提示“登录账号已存在！”。
 - 2) 另外还约束登录账号不能超过 20 位，登录账号不能为空。
 - 3) 如果密码少于 6 位，将提示“密码最少 6 位”。如果没有输入密码，将提示“密码不能为空。”
 - 4) 当确认密码输入与密码不同时，显示“两次输入密码不一致”。
 - 5) 姓名约束长度不能超过 20 位，没有输入姓名的状态下，姓名的默认为无名氏。
 - 6) 邮箱增加了格式校验功能，当邮箱为空时，提示“邮箱不能为空”。如果邮箱格式无效则提示“邮箱格式不对”。
 - 7) 电话号码约束最大长度不能超过 11 位。
 - 8) 点击头像时会打开本机文件，选择照片后即可上传为头像。
- 部分错误提示如图 5-2 所示。

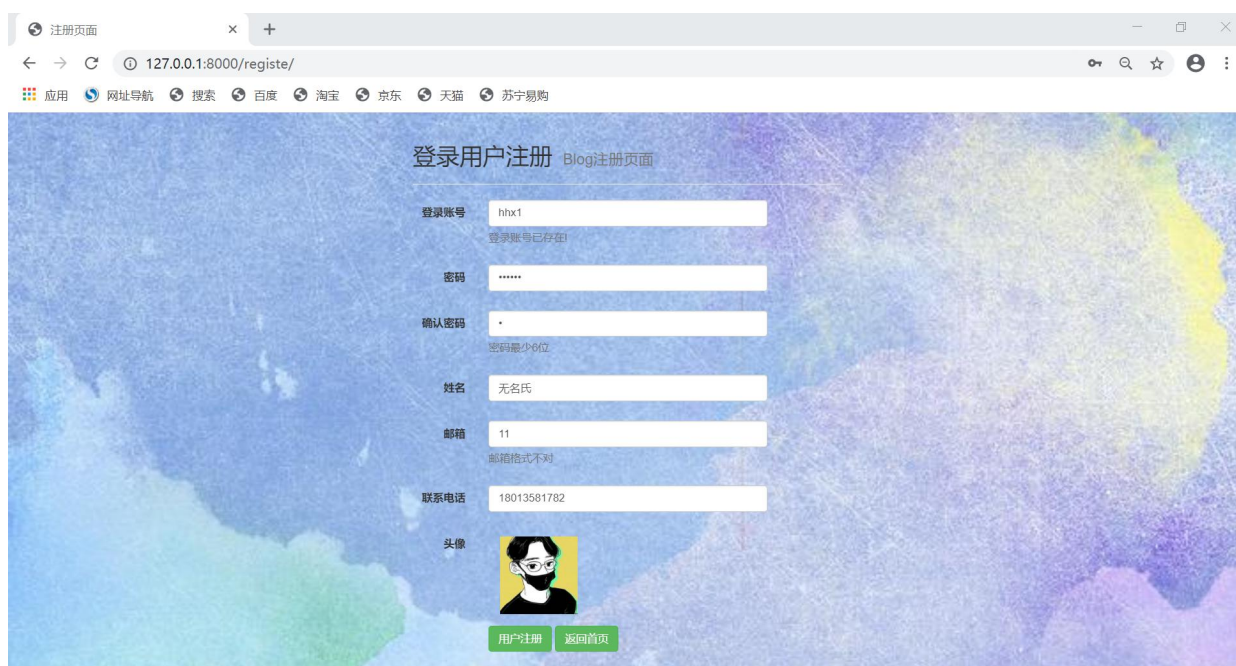


图 5-2 注册部分错误提示图

注册成功后会自动成为登录状态进入主界面。注册功能流程图如图 5-3 所示。

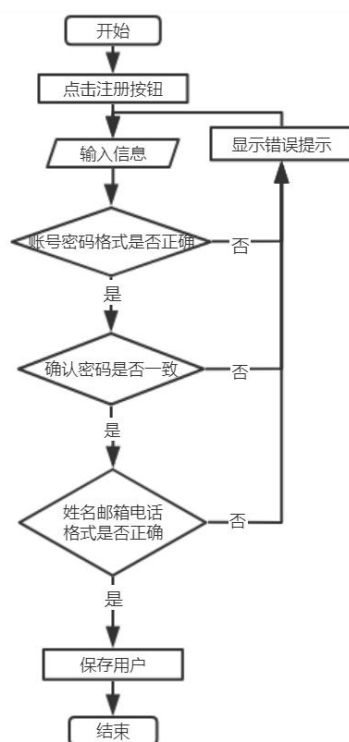


图 5-3 注册功能流程图

(2) 登录

输入注册时的用户名和密码，登录后将跳转到你的个人主页。

登录界面如图 5-4 所示。

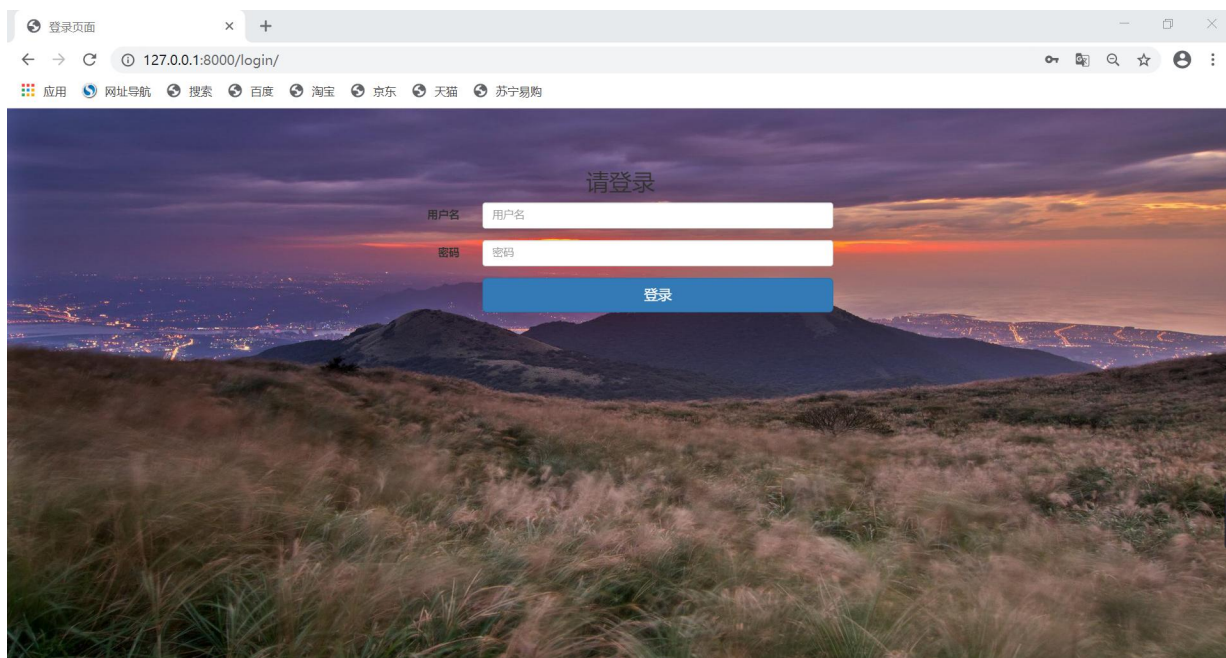


图 5-4 登录界面图

如果账号或密码输入错误则会显示错误提示信息，如图 5-5。

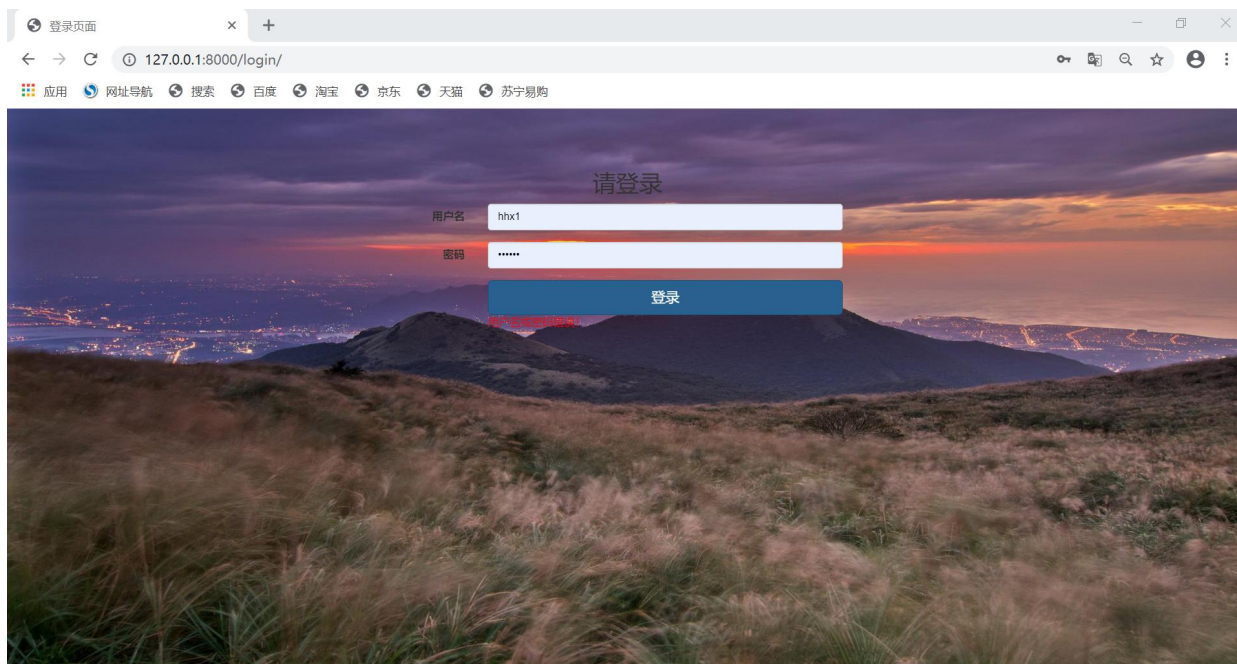


图 5-5 登录部分错误提示图

登录操作的流程图如图 5-6。

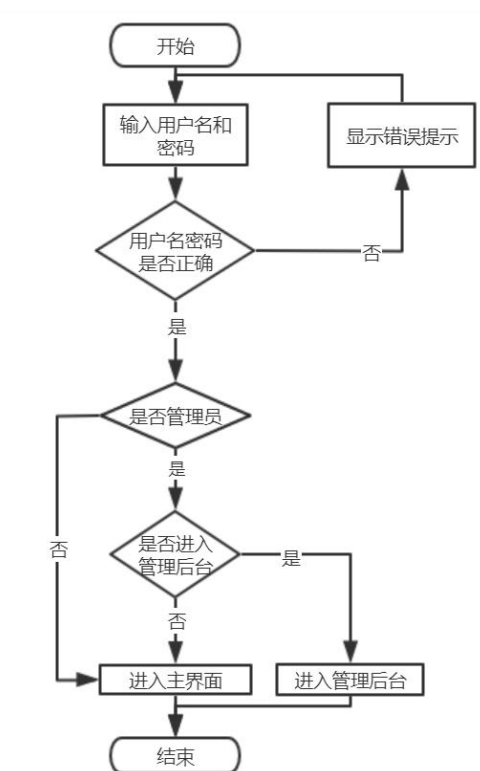


图 5-6 登录功能流程图

（3）个人资料

用户登录后导航条将会有个人资料栏，点击后将会进入到用户的个人资料卡界面，显示当前用户的姓名、邮箱、联系方式、账号、头像。用户资料卡界面如图 5-7。

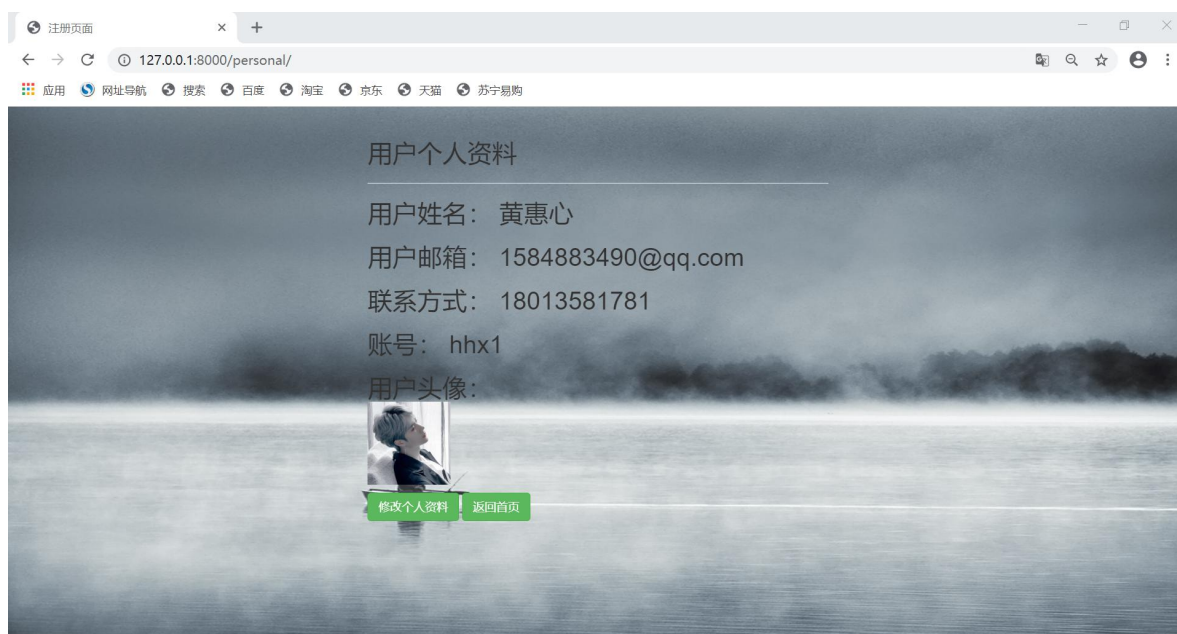


图 5-7 用户资料卡

点击资料卡的修改个人资料按钮，用户可以对自己的姓名、联系电话、邮箱信息进行更改，如图 5-8。



图 5-8 修改个人资料

（4）我的文章

导航条中有个“我的”链接，这个链接的主要功能是显示自己发布的文章。“我的”只有当用户登录后才会显示，展示效果如图 5-9。

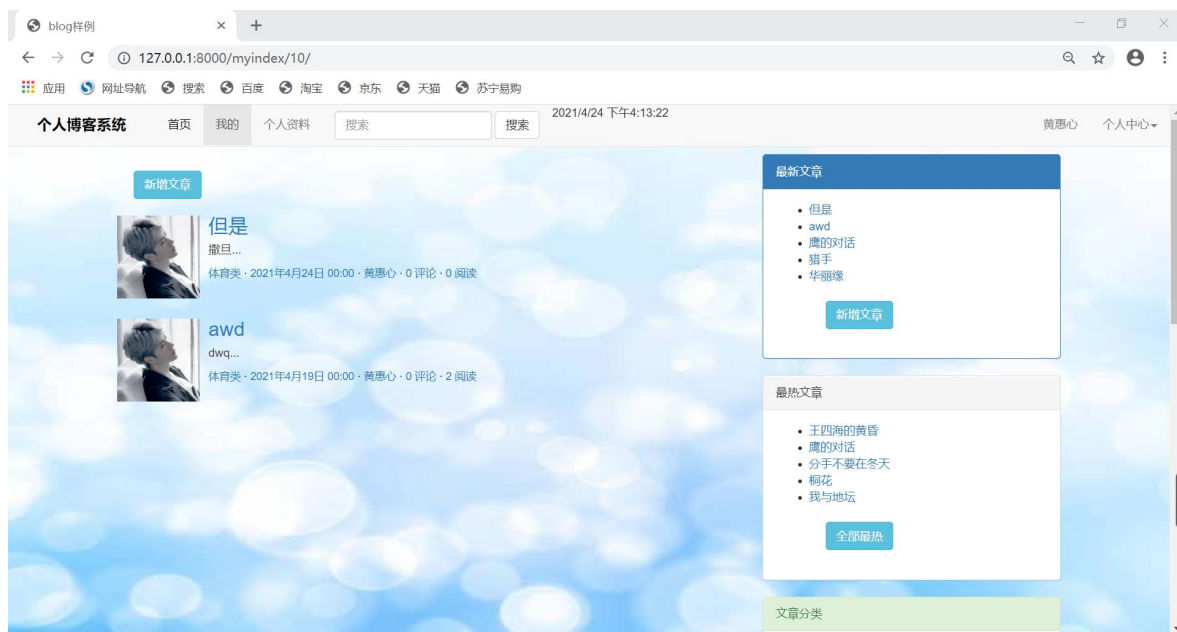


图 5-9 我的文章

5.1.2 文章管理模块

进入系统后进入到主界面，在主界面按照时间倒序浏览文章。界面具体如图 5-10。



图 5-10 博客系统主界面

不论任何权限的使用者，进入系统后就是博客的主界面。在游客状态下导航栏依次是首页、搜索框、时间显示、登录、注册，在登录状态下导航栏依次是首页、我的、个人资料、搜索框、时间显示、昵称显示、个人中心。博客首页的布局如下，第一行是导航条，中间主体部分是最新博文的链接，点击就可以跳转到相应的博文详细页面，博文数超过五个时进行分页。

导航条上有一个文本输入框，在输入框内输入字符串并点击搜索就能够检索出包括该字符串的文章，并且检索词在文章中标红显示，相当于一个简单的搜索引擎^[21]。如搜索“艺术”后搜索结果显示如图 5-11 所示。



图 5-11 搜索结果显示

首页每条博文的显示为，左侧显示文章作者的头像，并由上到下由左到右分别显示：文章的题目、文章简介、文章分类、文章的发表时间、文章作者、评论数、阅读数。这里设置了头像链接功能，点击头像将会显示所有此用户发表的文章，点击文章题目将会显示文章详情。

首页右部包括五个标签栏，分别是最新文章、最热文章、文章分类、文章标签、时间归档，另外首页还包括了一个音乐播放器，音乐播放器效果如图 5-12。

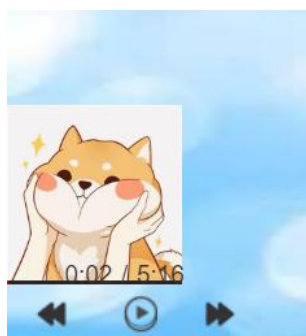


图 5-12 音乐播放器

登录后用户可以发布自己的博客，输入标题、摘要、内容，选择合适的分类和标签，发表成功后将在前台显示文章。写博客界面如图 5-13。

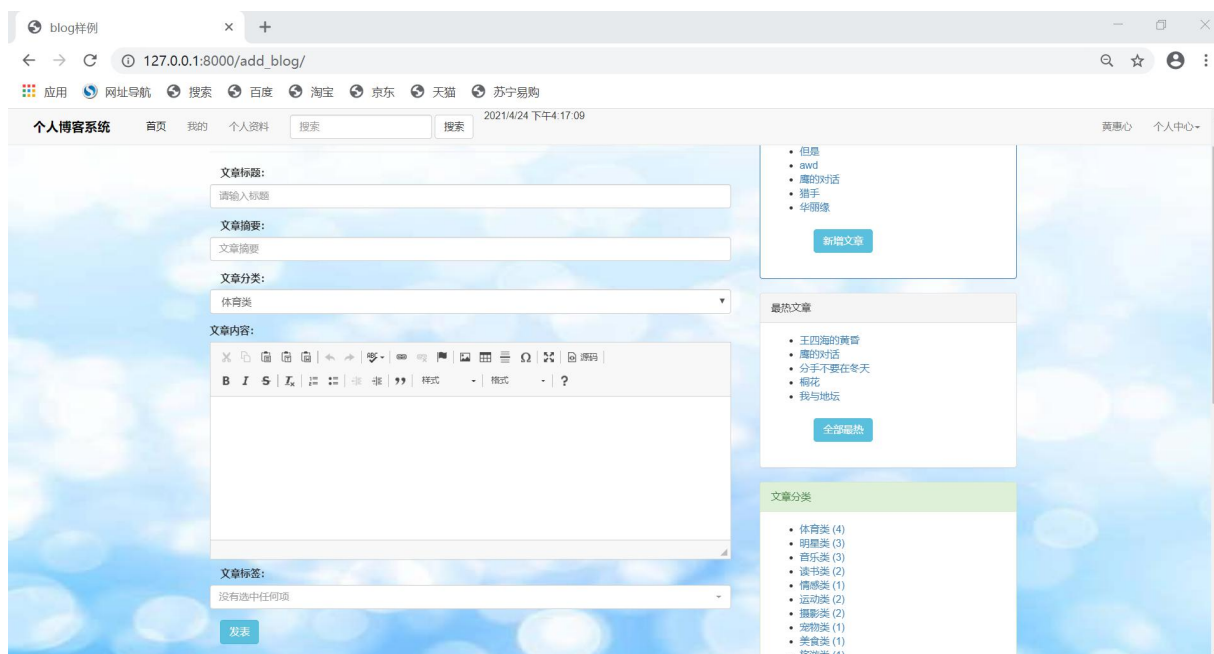


图 5-13 写博客界面

博客在发表时将会进行文章审查，主要方法是在数据库中创建一个敏感词表，在 add_blog 时判断文章内容中是否包含表中的词。审查结果显示如图 5-14。



图 5-14 审查结果提示

文章信息中包括阅读量，可以记录当前文章一共被阅读的次数：

```
views = models.IntegerField(default=0,verbose_name='查看次数')
```

每次点开文章详情，阅读量加一。

```
def increase_views(self):  
    self.views += 1  
    self.save(update_fields=['views'])
```

文章详情界面如图 5-15 所示。



图 5-15 文章详情界面

用户可以对自己的文章进行编辑，编辑博文页面如图 5-16 所示。



图 5-16 编辑文章界面

5.1.3 互动管理模块

登录后的用户可以在文章详情界面发表评论、删除评论，评论部分如图 5-17。

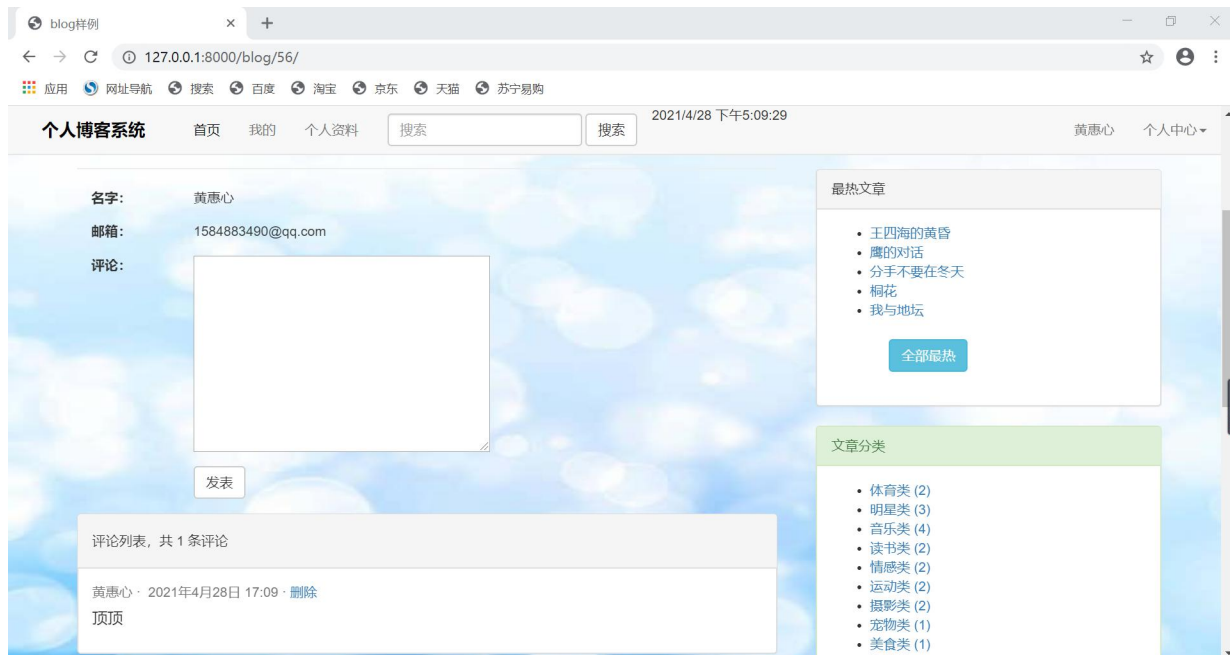


图 5-17 评论界面

5.2 系统后台管理模块的实现

系统后台只有管理员登录才能进入，系统管理员在后台管理系统各模块。博客系统后台界面如图 5-18 所示。后台管理基本流程图如图 5-19 所示，主要展示了管理员进入后台可进行操作的模块。

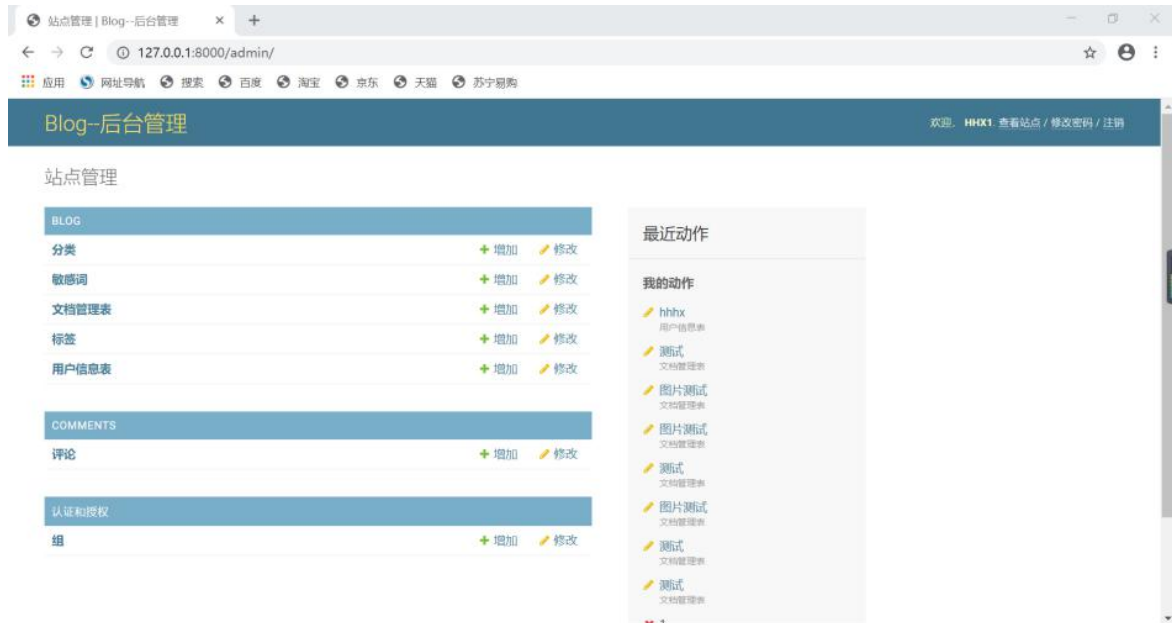


图 5-18 后台界面

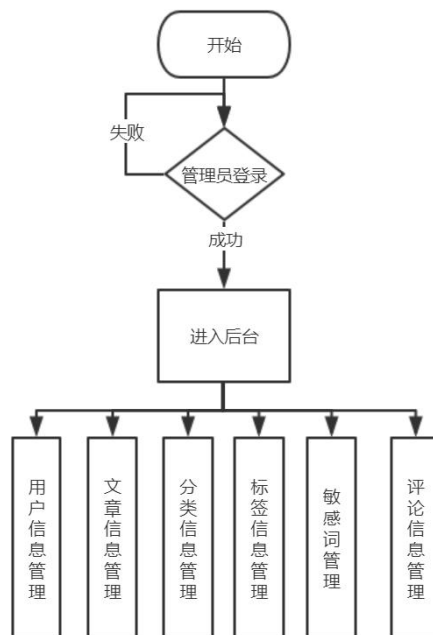


图 5-19 后台管理流程图

以用户管理为例，管理员可以修改用户信息，新增或删除用户，还可以设定用户权限。用户信息表如图 5-20 所示。

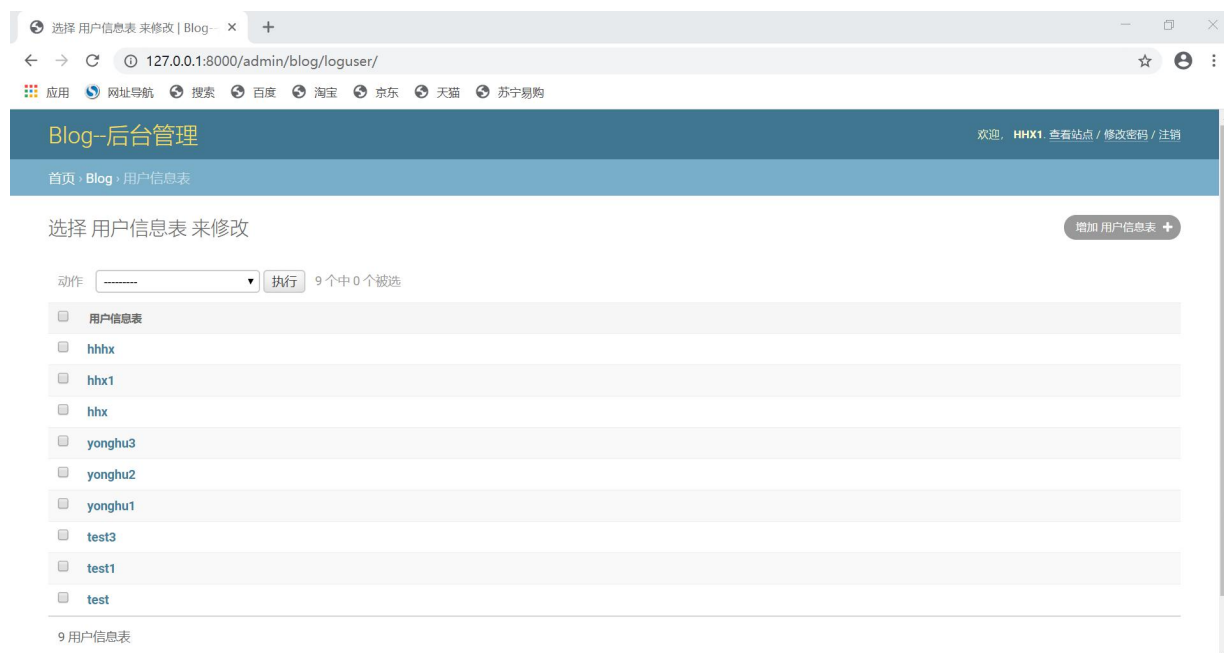


图 5-20 后台用户管理

此外，管理员在后台还可以对所有的博文信息进行管理，可以为博文选择分类、标签。分类主要用来划分博客类别，例如体育类、音乐类等。分类可以在后台增加删除和修改，每个分类有一个备注。标签用来标出文章所涉及到的相关领域，管理员可以对标签进行增加、修改和删除，可以为标签添加备注。敏感词是用来对发表的文章进行审查是否违规的标准，管理员在后台设置敏感词，要发表的文章将会对这些敏感词进行过滤。登录后的用户可以利用评论和其他人互动，管理员在后台管理评论信息。

6 系统测试

6.1 开发环境和系统部署

本系统使用 Python 作为开发语言，采用 Django 框架进行开发，页面开发使用 Bootstrap 框架，数据库选用 Django 内置的数据库 SQLite3，页面文件的编写和一些交互的实现也涉及 Html, JavaScript, 开发环境使用 Pycharm, 测试系统为 Windows 10, 另外还使用了富文本编辑器和 Pillow。具体开发环境及版本参数如下：

- (1) 开发工具 PyCharm 2020.1
- (2) 操作系统 Windows 10
- (3) 测试浏览器 Google Chrome 浏览器、Microsoft edge 浏览器、IE 浏览器
- (4) 系统框架 Django 2.2.4

系统部署：

(1) 到 Python 官方网站 <https://www.python.org/downloads/> 下载合适的 Python 版本，点击页面中的下载链接即可下载。

(2) 进入 Django 下载网站 <https://www.djangoproject.com/download/>，根据计算机系统选择下载合适的安装包，下载后按照提示安装。

(3) 在命令行终端输入以下命令安装 django-ckeditor。

```
pip install django-ckeditor
```

(4) 在命令行终端输入以下命令进行 pillow 的安装。

```
pip install pillow
```

(5) 在命令行输入以下命令安装 django haystack

```
pip install whoosh django_haystack jieba
```

(6) 通过 cmd+R 新建一个命令行窗口，切换至博客系统文件根目录。

运行命令：`python manage.py runserver`

运行成功的命令行显示如图 6-1，表示博客系统正在运行。

```
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
May 02, 2021 - 21:29:40
Django version 2.2.17, using settings 'test_blog.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

图 6-1 系统运行成功

打开浏览器输入 <http://127.0.0.1:8000/>即可进行访问测试。

6.2 测试

6.2.1 系统测试方法

本项目是一个 Web 网页的博客系统，因此测试是通过浏览器点击进行的测试，主要采用黑盒测试。本次测试通过运行程序并使用来查找程序中可能存在的问题，测试内容包括了系统中所有功能模块，并对系统的兼容性进行了测试。

6.2.2 功能测试

本节将根据上述测试方法对博客系统功能进行测试。测试内容应覆盖博客系统所有功能，因篇幅有限，在此选择典型测试用例予以说明。

（1）注册测试

测试内容：单击主界面的注册按钮看是否能跳转到注册界面，输入注册信息并提交后查看用户的注册信息是否存储到数据库中。

测试结果：点击提交后界面跳转到登录后的主界面，数据库中存入了用户的注册数据，注册成功。

（2）登录测试

测试内容：点击主界面右上方登录按钮，输入已经成功注册的用户名和密码，点击登录按钮观察能否成功登录。

测试结果：点击登录按钮跳转到登录后的主界面，登录成功。

（3）文章发表测试

测试内容：点击新增文章按钮，在编辑文章界面输入文章题目、内容等文章信息，点击发表观察能否发表成功。

测试结果：点击发表按钮后界面跳转到我的文章界面，看到刚刚输入的文章已经成功发表，测试成功。

（4）文章审查测试

测试内容：点击新增文章按钮，在输入文章内容时内容中包含敏感词，点击发表文章，观察文章能否发表。

测试结果：点击发表文章后浏览器弹出提示文章中包含敏感词，并请用户修改，点击确定后返回到新增文章界面，文章没有发表，测试成功。

（5）添加分类测试

测试内容：用户新增分类，观察分类是否成功添加，能否使用此分类发表文章。

测试结果：用户新增分类后，后台看到分类已成功添加进数据库，使用此分类发表文章，发表文章成功。

（6）发表评论测试

测试内容：进入一篇博客，输入要发表的文字，点击发表评论。

测试结果：观察到评论正常显示在博客下方，发表评论成功。

6.2.3 兼容性测试

（1）操作系统兼容性测试

在 Windows 7、Windows 10 和 Linux 系统下分别使用此博客系统，在不同操作系统下均可正常使用。

（2）浏览器兼容性测试

由于每个用户使用的浏览器是不确定的，因此要对该系统在不同浏览器中是否能够正常使用进行测试。

本次测试使用 Google Chrome 浏览器、IE 浏览器和 Microsoft edge 浏览器作为测试浏览器，观察能否正常登录网站、系统核心功能模块能否正常使用，得到在多种浏览器中系统均能正常使用，测试成功。

7 总结与展望

7.1 全文总结

本文基于软件工程学原理，使用 Django 开发了博客系统。本文先是介绍了开发该博客系统的背景，阐述了本项目的研究意义，介绍了国内外的研究情况，并介绍了开发本系统所使用的关键性技术。按照软件开发的基本流程对系统进行需求分析、系统设计、系统实现和测试。主要包括以下工作：

- （1）首先从不同功能模块对博客系统进行需求分析，并利用角色用例图从游客、普通用户、管理员的角度需求分析。
- （2）根据需求分析对博客系统进行详细设计，并利用流程图、功能结构图帮助完成设计。
- （3）完成系统模块的代码实现。
- （4）将博客系统在不同操作系统下和不同浏览器中测试，证明了博客系统基本运行正常。

7.2 展望

由于本文只是一个阶段性的设计实现，仍然有很多不足之处。

- （1）目前系统只是实现了一些基本功能，比如发表博客、评论博客、博客分类等，仍缺少一些更完善的功能，如关注博主、收藏博客、发送私信等。由于时间原因，目前还未能将功能全部完善，未来还将继续完善，将相关的辅助功能做的更好。
- （2）系统可以进一步增加博文推荐模块，根据用户的浏览喜好个性化推荐博文。
- （3）目前本系统还只能在浏览器上使用，如果想要更普遍的使用还需要考虑在移动端的使用。
- （4）目前系统的设计主要是以简洁为主，前端 UI 设计不太到位，没有更多的特效和复杂功能，后期可以完善交互性视觉内容。
- （5）后期应更重视信息安全，使用更安全的方案，增强系统稳健性。

尽管系统有很多不足，但展望未来，解决了这些问题之后，本系统可以成为较完善的可以使用的网站。希望本文对于基于 Django 的博客系统开发能够做出一点贡献。

参考文献

- [1] 邹竞莹.Node.JS 博客系统的设计与实现[D].黑龙江:黑龙江大学,2016.
- [2] 安东尼奥·米勒.Django 项目实例精解[M]. 北京:清华大学出版社, 2019.
- [3] Jeremy Nelson. Building a Library App Portfolio with Redis and Django[J]. Code4Lib Journal,2013(19).
- [4] 刘立行,张诗.视频博客阅听众使用行为研究之初探[J].东南传播,2020,(11):45-49.
- [5] 雷霄.浅析现代网络趣缘群体对于媒体平台的使用研究——以轻博客平台” 网易 LOFTER” 为例[J].数字通信世界,2018,(11):243-244.
- [6] 季渝.论微博作品的著作权保护[J].法制与社会,2020,(27):33-34.
- [7] 王琛.学术博客影响力评价研究——以科学网博客为例[D].山西:山西财经大学,2018.
- [8] 佟文娟,宁健铭.微博客对新闻传播的影响[J].新闻传播,2017,(7):91,93.
- [9] 常佳宁,李阳齐.基于 Django 的个人博客系统设计开发[J].中国科技信息,2021,(2):75-77.
- [10] 涂远杰,郑剑.基于 Flask 的博客网站设计与实现[J].电脑知识与技术,2020,16(15):109-111.
- [11] 李嘉明.基于 Node.js 多人博客系统的设计与实现[J].电脑知识与技术,2020,16(9):71-72,75.
- [12] 鞠宏军,林涛.基于 Spring Boot 的博客系统的设计与实现[J].电脑知识与技术,2019,15(33):50-52,60.
- [13] 谢建华,梁杰华,郑剑.基于 Django 实现四方博客[J].电脑知识与技术,2019,15(23):51-54.
- [14] 江柳.基于 Django 的博客系统开发研究[J].电脑编程技巧与维护,2016(13):14-15.
- [15] 陈绪.基于 MongoDB 数据库的博客管理系统[D].河北:河北农业大学,2018.
- [16] 董云影,张红.基于 Python 的博客设计[J].科学与财富,2019,(5):76.
- [17] 刘志凯,张太红.Django 框架在 web 开发中的应用[J].农业网络信息,2015,(2):51-52.
- [18] 杨志庆.基于 Django 的 Blog 系统的开发与实现[J].机电一体化,2013,19(9):69-72.

- [19] 张晓.Django 实战[M]. 北京:人民邮电出版社, 2020.
- [20] 罗涛. 基于 Spring Boot 的多用户博客系统的设计研究[D]. 青海:青海师范大学,2020.
- [21] 邓笑. 基于 Spring Boot 的校园轻博客系统的设计与实现[D]. 湖北:华中科技大学,2018.

致谢

随着本篇论文结束，我意识到我的本科生涯也将在此画下句号。很荣幸能够在苏州大学完成我的本科学习，回忆我的大学四年，有收获有快乐也有遗憾。今后我一定会常回我的母校看看，永远记得大学带给我的成长。

在撰写毕业论文的过程中，我体会到重要的不仅仅是编程能力，更重要的是掌握正确的流程和方法，还有在开发过程中与老师同伴的沟通。这些收获不仅会在毕业设计的过程中帮助我，也使我的学习和生活都受益良多。

感谢我的指导老师韩冬老师，在我的毕业设计过程中尽心尽力的指导，帮助我解决问题，为我提供建议指明方向。感谢计科院的各位老师，是在各位老师的培养下我才从对编程一无所知逐渐进步，感谢老师的无私奉献和付出，使我不仅学会知识技术也学会了如何做人。感谢我的父母对我的关爱，给了我这么好的学习平台，没有他们我无法完成我的本科学业生活。感谢我的同学朋友，他们在大学四年陪伴了我，在这四年中带给了我快乐，给与了我帮助。

最后，我会在以后的日子里，戒骄戒躁，更加严格要求自己，谨记苏州大学和各位老师对我的教诲。

苏州大学本科生毕业设计（论文）任务书

学院（部）：计算机科学与技术学院

设计（论文）选题：基于 Django 的博客系统开发					
指导教师姓名	韩冬	职 称	副教授	类 别	毕业设计
学 生 姓 名	黄惠心	学 号	1727405129	设计（论文）类型	应用型课题
年 级	2017	专 业	计算机科学与技术	是否隶属科研项目	否
<p>1、设计（论文）的主要任务及目标</p> <p>搭建一个基于 Django 框架的个人博客系统。博客系统的基本功能包括用户注册和登录、用户管理、发表博文、删除博文、评论以及搜索博文等。</p> <p>基于 Django 的博客系统开发设计中，使用 Pycharm，主要使用的是 python 语言，同时，页面文件的编写和一些交互的实现也涉及 Html, JavaScript。技术方面，使用的 python 中的稳定框架 Django 进行开发，在选用数据库时，选用 Django 内置的数据库 SQLite3。</p>					
<p>2、设计（论文）的主要内容</p> <p>个人博客系统的可行性和需求分析主要包括：文章管理功能、用户管理功能、评论管理功能、文章标签功能。分析博客系统的总体设计主要包括：系统的架构设计、系统的功能设计、数据库设计。</p> <p>博客系统的开发工具分析：该博客系统采用 Django 框架，Python 语言技术进行开发。</p> <p>利用 Django 开发博客系统包括以下几块内容:利用 Django 的管理功能创建项目模版、超级用户、设计数据库模型,开发前端页面和视图函数,并配置 url 和视图函数的映射规则,进行博客访问。</p> <p>通过分析 Django 框架结构及其特点并以一个简易的博客系统的创建，展示 Django 在 Web 开发中的强大优势。实践证明，使用 Django 可以降低 Web 应用开发的复杂性，提高开发效率。</p>					

3、设计（论文）的基本要求

根据任务书的要求把控好整体进度，至少每 2 周在毕业设计（论文）管理系统内提交一次进展情况报告。

应独立翻译一篇与毕业设计（论文）工作相关的外文文献，参阅外文文献资料一般不得少于 3000 个单词（字）。

根据自身对毕业设计（论文）所选题目的总体认识、文献资料查阅情况及毕业设计（论文）进度规划等，在毕业设计（论文）工作开始 2 周内，提交一篇不少于 1500 字或同等学术话语外文单词数（摘要和参考文献不计入总字数）的文献综述报告。

毕业设计（论文）的撰写应做到条理清晰，逻辑性强，符合写作规范。正文字数（不含封面、目录、中英文摘要、参考文献、致谢、各类附录等）不得少于 8000 字。

毕业设计（论文）须在管理系统内提交查重，可参加答辩的毕业设计（论文）查重报告中“总文字复制比”不得超过 25%；推荐为校级优秀毕业设计（论文）的“总文字复制比”不得超过 15%。

4、主要参考文献

- [1] 张晓贵编；张天怡. Django 实战[M]. 北京：人民邮电出版社. 2020.
- [2] （美）安东尼奥·米勒著；李伟译. Django 项目实例精解[M]. 北京：清华大学出版社. 2019.
- [3] 常佳宁, 李阳齐. 基于 Django 的个人博客系统设计开发[J]. 中国科技信息, 2021(02):75-77.
- [4] 江柳. 基于 Django 的博客系统开发研究[J]. 电脑编程技巧与维护, 2016(13):14-15.
- [5] 谢建华, 梁杰华, 郑剑. 基于 Django 实现四方博客[J]. 电脑知识与技术, 2019, 15(23):51-54.
- [6] 董云影, 张红. 基于 Python 的博客设计[J]. 科学与财富, 2019, (5):76. DOI:10.3969/j.issn.1671-2226.2019.05.072.
- [7] 刘志凯, 张太红. Django 框架在 web 开发中的应用[J]. 农业网络信息, 2015, (2):51-52. DOI:10.3969/j.issn.1672-6251.2015.02.014.
- [8] 杨志庆. 基于 Django 的 Blog 系统的开发与实现[J]. 机电一体化, 2013, 19(9):69-72. DOI:10.3969/j.issn.1007-080x.2013.09.013.
- [9] Jeremy Nelson. Building a Library App Portfolio with Redis and Django[J]. Code4Lib Journal, 2013(19).
- [10] 鞠宏军, 林涛. 基于 Spring Boot 的博客系统的设计与实现[J]. 电脑知识与技术, 2019, 15(33):50-52, 60.
- [11] 陈绪. 基于 MongoDB 数据库的博客管理系统[D]. 河北:河北农业大学, 2018.

5、进度安排		
	设计（论文）各阶段任务	起 止 日 期
1	收集有关资料，写任务书	2021-01-23 至 2021-02-05
2	确定实施方案，外文翻译	2021-02-06 至 2021-02-17
3	熟悉开发环境，文献综述	2021-02-17 至 2021-02-27
4	程序设计阶段，论文草稿	2021-02-27 至 2021-04-10
5	撰写毕业论文，论文定稿	2021-04-10 至 2021-05-01
6	准备答辩阶段	2021-05-02 至 2021-05-12

- 注：1、此表一式三份，学院(部)、指导教师、学生各一份；
2、类别是指毕业论文或毕业设计，类型指应用型、理论研究型和其他；
3、在指导教师的指导下由学生填写。

Building a Library App Portfolio with Redis and Django

The Tutt Library at Colorado College is developing a portfolio of library applications for use by patrons and library staff. Developed under an iterative and incremental agile model, these single-use HTML5 applications target multiple devices while using Bootstrap and Django to deliver fast and responsive interfaces to underlying FRBR datastores running on Redis, an advanced NoSQL database server. Two types are delineated: applications for access and discovery, which are available to everyone; and productivity applications, which are primarily for library staff to administer and manage the FRBR-RDA records. The access portfolio includes Book Search, Article Search, Call Number, and Library Hours applications. The productivity side includes an Orders App and a MARC Batch application for ingesting MARC records as FRBR entities using RDA Core attributes. When a critical threshold is reached, the Tutt Library intends to replace its legacy ILS with this library application portfolio.

By Jeremy Nelson

Background

Colorado College's Tutt Library is a small, urban, academic library serving the needs of over 2,000 students along with faculty and staff in Colorado Springs, Colorado. As a member of the Colorado Alliance of Research Libraries, a consortium of academic and public libraries in Colorado and Wyoming, we participate in a union catalog comprising the collections of our member institutions. Our Islandora Institutional Repository is also hosted through the Colorado Alliance's repository service. We operate our own instance of III's Millennium ILS, however. Like most academic libraries, our material budgets have shifted from physical to electronic resources and, as a consequence, we are doing more batch loading of MARC records for these electronic resources and less original or copy-cataloging of print material.

With the quality of vendor-supplied MARC records varying considerably, the old workflows to manipulate and load records into our legacy ILS were often long and laborious processes. By scripting the manipulation of these MARC records with Python and the pymarc Python module [1], and with a web front-end built in Django, we brought

considerable time savings to our cataloging staff. This led to a second Django project that enabled senior students to self-submit their thesis or final essay, along with any supporting datasets, to our digital repository.

A parallel effort was started as we looked at various commercial and open source options for a new discovery layer. Given monetary and resource constraints, along with the worry of maintaining multiple codebases in different programming languages and environments, the library decided to fork Kochief, a Django-based discovery project. This decision led to the release of Aristotle, the Tutt Library discovery-layer project available on Github under the Apache 2 open source license [2].

In 2011, we started to explore using Redis, a popular NoSQL technology, to represent bibliographic and operational information. This research led to the FRBR-Redis datastore project that was the topic of a 2012 Code4Lib presentation [3]. Using the Library of Congress Bibliographic Framework Initiative's "A Bibliographic Framework for a Digital Age" [4] as starting, high-level requirements for the FRBR-Redis datastore, we tested the suitability of using Redis with over 850 unit tests representing MARC21, MODS, and VRA metadata structured as FRBR Work-Expression-Manifestation-Item entities with a variety of Redis data primitives.

One of the trends highlighted at the 2012 Code4Lib conference was the growing importance of mobile devices, both in usage by library patrons, and in the limitations and opportunities presented by developing applications. This caused us to rethink how we offered access and management to bibliographic information from a desktop/rich web "discover" model to a simplified mobile app user interface used in mobile and tablet native applications such as Apple's iOS and Google's Android operating system. We started experimenting with Twitter's Bootstrap support for responsive web design to build simplified HTML5 apps that were much closer to the design aesthetic of a mobile and tablet application. The first app we released was a call number app that provides an embedded widget for our discovery layer as well as a standalone app targeted for mobile and tablet devices but fully usable by modern desktop web browsers. This app was the beginning of the Aristotle Library Apps project [5] and can viewed online at http://discovery.coloradocollege.edu/apps/call_number/.

Since 2011, we continued to monitor the modeling work done by the Bibliographic Framework Transition Initiative. As we became more experienced with Redis, the initial

implementation of a Redis FRBR datastore, with a single Redis instance using multiple databases, wasn't as flexible as having each each of the first group of FRBR entities (Work, Expression, Manifestation, and Items) run as separate Redis instances on different ports on the same server. In September 2012, Sally McCallum, in an IGeLU presentation [6], offered the first glimpse of a new bibliographic model she referred to as MARCR, for MARC Resource. This new model supports RDA and FRBR but the core entities have been whittled down to four: Work, Instance, Annotation, and Authority. Because of Redis's flexibility, we were quickly able to modify our existing FRBR Redis datastore key structure to follow this new model using RDA. A primer on this new model was published in November of 2012 [7], with the name changed to BIBFRAME and minor changes made to McCallum's introduction from September. Namely, the Work entity became Creative Work and the namespace for the model was formally set as BIBFRAME. The BIBFRAME Redis datastore is now a separate open source project licensed under the GNU General Public License 2 and includes documentation, Redis server configuration files for each of the BIBFRAME entities, and LUA server-sides scripts [8].

More about Redis

Redis, a key-value datastore, is one of the many new NoSQL data technologies that offer alternative models for data representation and use. Redis supports data persistence in two ways: an RDB mode that saves the dataset at periodic intervals, and an AOF mode that saves the dataset with every write operation. While there are advantages and disadvantages to each approach [9], most libraries could employ a combination of RDB mode for bibliographic records and AOF mode for library transactional data like circulation statistics.

Redis is fundamentally different from the flat-file structure of a MARC record and the relational databases of more traditional library systems. The flexibility of Redis allows for the rapid development of multiple apps by supporting different information schemes and structures within a single datastore. The manner in which a Redis key is constructed allows for the embedding of semantic or heuristic information about the data structure in its naming structure. Redis assumes that related data use a key naming pattern, and even provides a global function to increment the key ID.

Another important design consideration when using Redis is the type of data primitive to associate with a key. The simplest value is an atomic string. The Redis list is a collection primitive which stores unordered and duplicate string values. The Redis set stores unique string values, while the sorted set assigns a sort weight to each value. If a weight of 0 is used in a sorted set, Redis does a lexical sort based on the string values in the set. The last Redis data primitive is a hash. A Redis hash associates multiple sub-keys with a single Redis key and with the HGET command returns the value associated with that sub-key.

Redis is not a relational database and it would be suboptimal to attempt to replicate an RDBMS. In Redis, the key is the fundamental structure, not a table-row as in an RDBMS. Redis keys can also serve as a string value for other keys in the datastore, providing a sort of crude SQL JOIN, but offering more flexibility in representing relationships between keys in a manner that would be difficult or impossible to replicate in an RDBMS. The downside is that referential integrity between different tables is not built into Redis. Eventual consistency can be achieved either through application logic or through strategies involving a combination of Redis server commands.

The Redis string, set, sorted set, list and hash data primitives all offer different ways to represent library information in the Redis server. Redis also provides a number of server and primitive-specific commands that ease application development, including EXIST and TYPE. For the EXIST command, a string is passed in as a parameter and a boolean is returned confirming whether the string is a key in the datastore. The TYPE command, when passed in a key string, returns the type of Redis data structure that is represented by the key or a null value if it doesn't exist.

For large datasets that may not fit into RAM, the lead developer on the Redis project, Salvatore Sanfilippo, recommends presharding [10] to break up the datastore among different Redis instances. The naming schema used by the Aristotle Library Apps and BIBFRAME Datastore project can easily support presharding for use in large collections, though for the time being, a single Virtual Machine with 4G of RAM is more than sufficient to run all of Colorado College's BIBFRAME Datastore Redis instances. The following graphic shows our Colorado College BIBFRAME Datastore set-up.

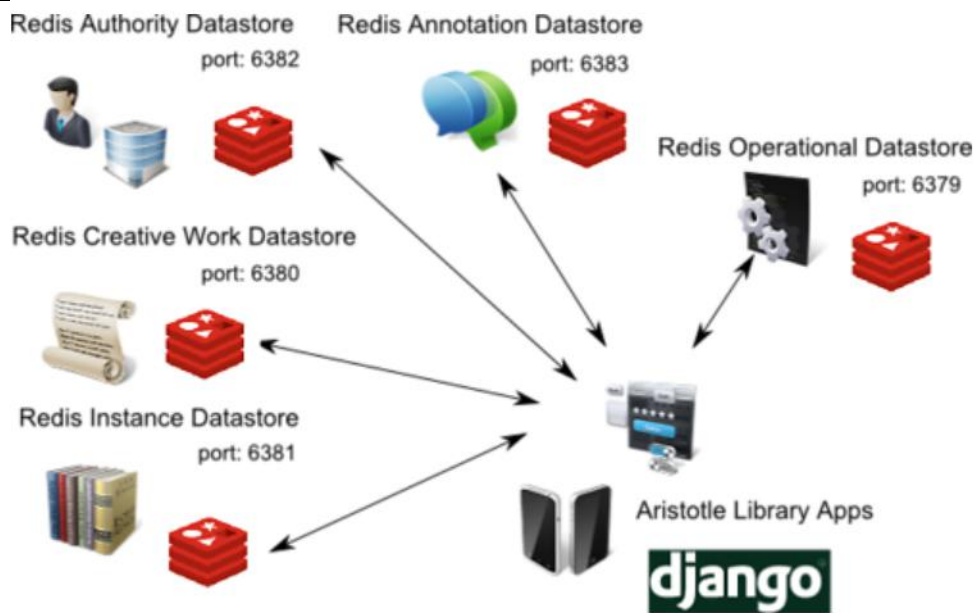


Figure 1. Colorado College's BIBFRAME Datastore Setup

The Evolution of a Native Bibliographic Redis Datastore

The first iteration of a Redis bibliographic datastore used a naming schema based on FRBR and RDA, using a notation similar to XML namespaces to create a collection of keys to represent values and relationships with other entities in the datastore.

For example, in the Aristotle Library Apps project, RDA Core FRBR entities and attributes are extracted from MARC records following the MARC-to-RDA mappings provided in the ALA's RDA Toolkit [11] that are then organized into Redis key collections following a pattern that maps to BIBFRAME linkable information resources like Creative Work, Instance, Authority, and Annotation. The figure below shows one such BIBFRAME Redis key collection for a CreativeWork's RDA:Title with supporting Redis keys for that entity.

The phonetic, `rda:variantTitleForTheWork:sort` subkeys and subvalues in the `bibframe:CreativeWork's rda:Title` support Title searches by various apps and Colorado College's Discovery Layer.

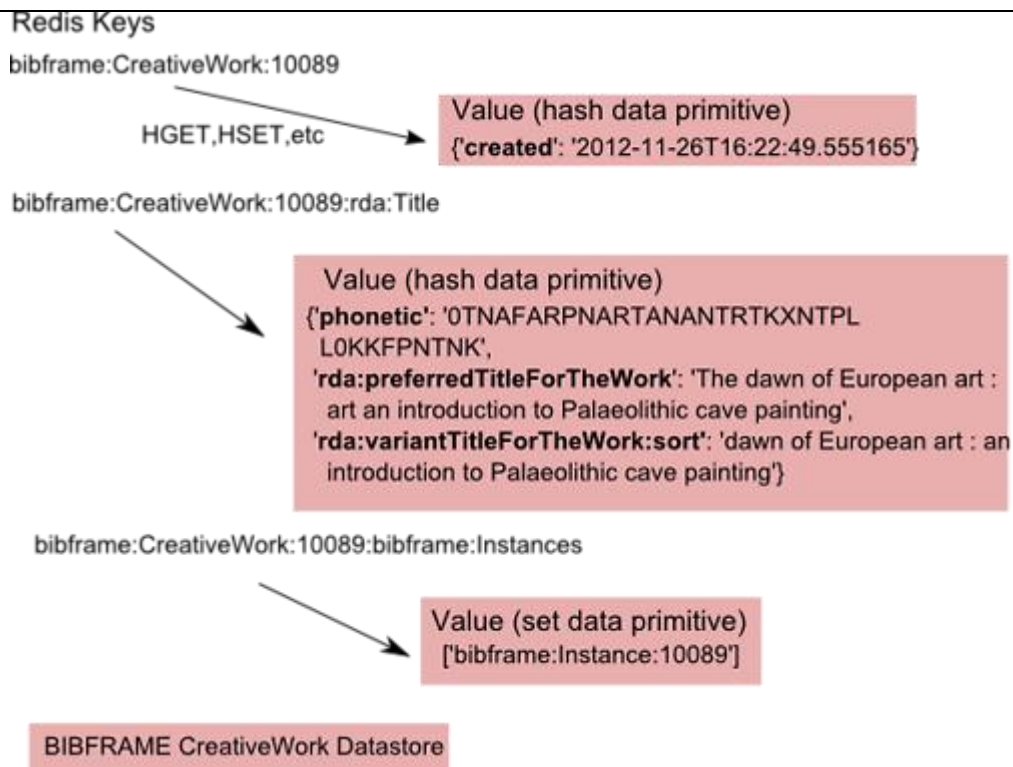


Figure 2. Example of Bibliographic Redis Keys and Data Primitive Values for a BIBFRAME Creative Work's Title

The BIBFRAME Datastore can support other metadata formats and values as well. Continuing the previous example, if we wanted to use MODS or Dublin Core instead of RDA, the BIBFRAME Datastore key could look like `bibframe:CreativeWork:10089:mods:titleInfo` with each of the MODS titleInfo sub-elements as hash key-value pairs for an individual CreativeWork. Likewise, a Redis key like `bibframe:CreativeWork:10089:dc:title` could be either a simple static string or a HASH like the current `rda:Title` key used in the current iteration of the BIBFRAME Datastore project. We could even have all three (or as many different keys as desired) exist in the same Redis datastore instance. The design of the Redis key's structure is through convention and convenience with an effort to be explicitly descriptive of the data that is returned by Redis without being too verbose.

Meanwhile, financial information, such as material orders and invoice information stored in our ILS, is extracted and added to the Redis datastore for reporting and budget forecasts. We even store the library hours as a Redis string for use in a standalone app and as a JSON data feed for our discovery layer.

Redis datastore interactions are abstracted via Python classes, which are built with the

redis-py module [6]. If the app developer needs custom data storage or extended Redis functionality, Python custom classes can extend existing classes through direct manipulation of the datastore.

HTML5, Responsive Web Design, and Twitter Bootstrap

While native apps generally run faster and more closely follow the recommended user interface guidelines for their respective platforms, the Tutt Library does not have the resources to maintain multiple app development environments. Thankfully, with evolving web standards and protocols, coupled with the development of CSS and JavaScript libraries, fast and easy-to-use HTML5 apps can be created that are more universal and can run on multiple mobile and tablet platforms as well as on personal computers running more modern web browsers that support HTML5. This is the goal of responsive web design, as expressed in the original article on A List Apart:

Rather than tailoring disconnected designs to each of an ever-increasing number of web devices, we can treat them as facets of the same experience. We can design for an optimal viewing experience, but embed standards-based technologies into our designs to make them not only more flexible, but more adaptive to the media that renders them. In short, we need to practice responsive web design. [7]

The Aristotle Library App project uses the popular web framework Bootstrap as the basis for the user interfaces that respond and adjust for different client devices and displays. It is prohibitively expensive and impossible for a small library with limited staff and resources to test out apps on all of the different platforms, web browsers, and devices used by our users. By focusing on the most popular and available devices in the library (Windows 7, Macintosh, iOS, and some Android phones and tablets), the Tutt Library targets specific functionality needed by its patrons and staff. The design intention of this HTML5-based app development environment is that creating a new app should be roughly equivalent in difficulty to building a simple website leveraging librarian and staff's pre-existing competencies with such tools as Dreamweaver and content management systems. While there is training involved in educating staff about Bootstrap and HTML5, the training burden and requirements for app development is considerably less than if the library tried to develop native apps for the iOS and Android environments.

Access and Discovery Apps

The majority of apps in the initial Aristotle Library App project are categorized as Access and Discovery Apps, which allow users to find and access the library resources and more general information about the library. Access and Discovery Apps broadly address the generic tasks by users to find, identify, select, and obtain resources as expressed in the FRBR specification [14]. Also included in this category, the Tutt Library's Hours App informs a patron if the library is open or closed. While not bibliographic in nature, the Hours App addresses one of the top questions the Tutt Library receives from patrons.

The Call Number App

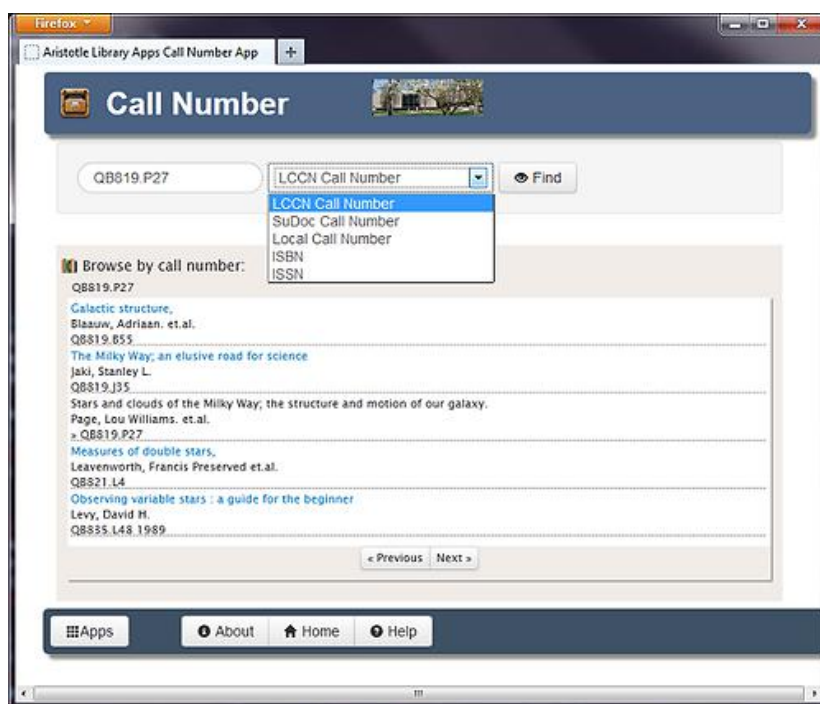


Figure 3. Screenshot of the Call Number App

The Call Number App was the first app released, and served as the catalyst for the entire Aristotle project. As we worked on the discovery layer, another librarian was inspired by a feature in Stanford University's Searchworks (built with Blacklight and Hydra) that allowed a patron to see which call numbers were near each other in the library's stacks. While investigating Stanford's implementation based on Solr, we realized a simplified data model could be used with Redis. To create the type of sorted indexes needed for this app, normalized Library of Congress, SuDoc, and local call numbers were added as weights to Redis sorted sets. Once we had embedded the Call Number App into the discovery layer, we explored the further development of dedicated, simplified apps for common searches. These independent apps could be used in larger systems, like the

discovery layer or the library's website, through the use of JSON APIs and raw HTML.

Library Hours App

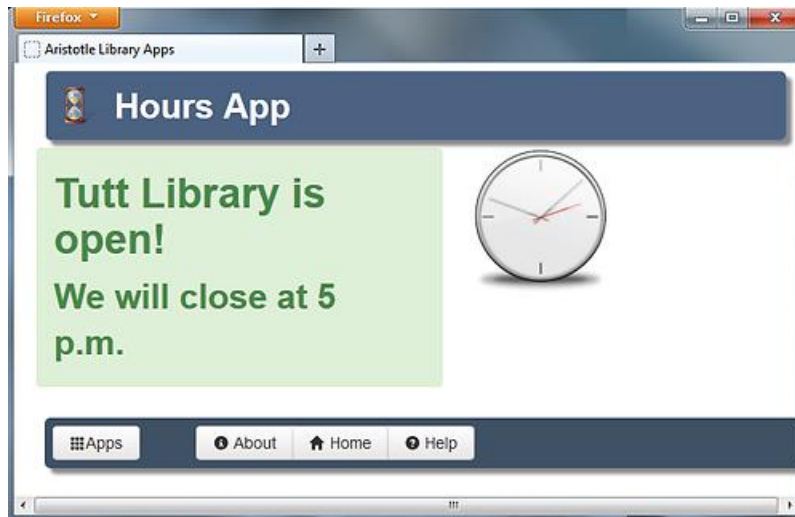


Figure 4. Screenshot of Library Hours App

When the college adopted a CMS incapable of building a dynamic feed of the library's hours of operations for the library's homepage, we felt that a dedicated app with a JSON feed and hours data stored in Redis would work instead. The Library Hours App stores dates and hours in a string using Redis bit operations commands. Each day has a unique key with the following pattern: "library-hours:YYYY-MM-DD", with the value being a 96-bit string with each bit representing a quarter hour with bit offset 1 representing 00:00 to 00:14, bit offset 2, 00:15-00:29, etc. for each quarter hour of a twenty-four hour day. Bits set to zero (the default) means the library is closed, the bit set to 1 means the library is open for that quarter hour.

To see if the library is open, the Hours App queries the library's operational Redis datastore by using a Redis GETBIT command as demonstrated in this code snippet from the Hours App's `redis_helpers.py` [15]:

```
def is_library_open(question_date=datetime.datetime.today(),
                    redis_ds=redis_ds):
    """
    Function checks datastore for open and closing times, returns
    True if library is open.
    <br/>
    :param question_date: Datetime object to check datastore, default
```

is the current datetime stamp.

```
:param redis_ds: Redis datastore, defaults to module redis_ds
:rtype: Boolean True or False
"""

offset = calculate_offset(question_date)
status_key = question_date.strftime(library_key_format)
return bool(int(redis_ds.getbit(status_key, offset)))
```

The patron user interface for the Hours App displays a simple message with the library's current hours. If the library is closed, the app displays the next available date and time when the library is open. With 365 keys per year, one for each day, this Redis structure easily supports the requirements of the Hours App administrative user interface, allowing authenticated library staff to add or modify the posted hours.

Productivity Apps

The second category of apps is for productivity, developed to either manage or report on resources in the collections. These apps require the user to first authenticate, then, depending on the app and the user's authorizations, allow for the manipulation or reporting of library information, which includes the native BIBFRAME entities in the datastore. In the Orders App, order records were imported from Tutt Library's legacy ILS into the FRBR Redis datastore. By doing so, we freed this information from the proprietary ILS vendor that tightly binds order information to the MARC bibliographic record (even going so far as to create custom 9xx fields for order information). By separating the order information into Redis sets with each invoice and order as distinct Redis keys, visualizations and budget reporting became much simpler. Before we had this tool we would have to export this data from the ILS and extract the data from the MARC21 record, then clean it up before importing it into Microsoft Excel for analysis of this critical aspect of library operational information.

We have also developed a productivity app for a collection of Fedora Commons utilities used by library staff to move objects around in the digital repository, ingest batches of objects, and apply a metadata batch update to one or more Fedora objects. While this app does not directly use the BIBFRAME entities, it has streamlined the workflow of staff in the library who work with the digital repository. The Aristotle Library Apps project is

flexible enough to accommodate workflows with other library systems, like our Fedora Commons digital repository.

Roadmap for the Aristotle Library App Project

The Tutt Library uses its Call Number, Hours, and Discovery apps to augment the library's website and discovery layer. These are publicly available, along with the Article and Book Search Apps, at <http://discovery.coloradocollege.edu/apps/>. The same JSON interface that the Call Number App uses to populate a shelf-browser is also used in the record view in the discovery layer. The Hours App provides an embedded HTML snippet for inclusion in various locations in the library's website.

The next wave of app development will focus both on improving local workflows for such library services as material check-out, course reserves, and collection inventories, as well as improved discoverability of resources from other institutions through shared BIBFRAME Redis datastores. In keeping with an agile software development philosophy, each app should be simple enough to design, implement, and start testing within a three-to-four week sprint, which nicely coincides with the current academic block calendar at Colorado College.

Peer-to-Peer and Consortium Union Catalogs

Concern over interoperability was brought up as a challenge to any radical technology change as the Tutt Library moved to an app model. The regional Colorado Alliance of Research Library's Prospector union catalog, in which Colorado College is both an active lender and borrower, supports the college's block plan, where students take intense, 3 1/2 week courses for college credit. With students needing research material promptly, Tutt Library strives to deliver to students, faculty, and staff as promptly as possible, preferably under 72 hours. The Prospector-based ILL service is critical to meet this tight deadline for materials. Any replacement or legacy ILS cannot diminish that service. Our plan to tackle these challenges involves an incremental approach, starting off with BIBFRAME datastores shared by just two institutions before increasing the scale to a consortium or regional scale.

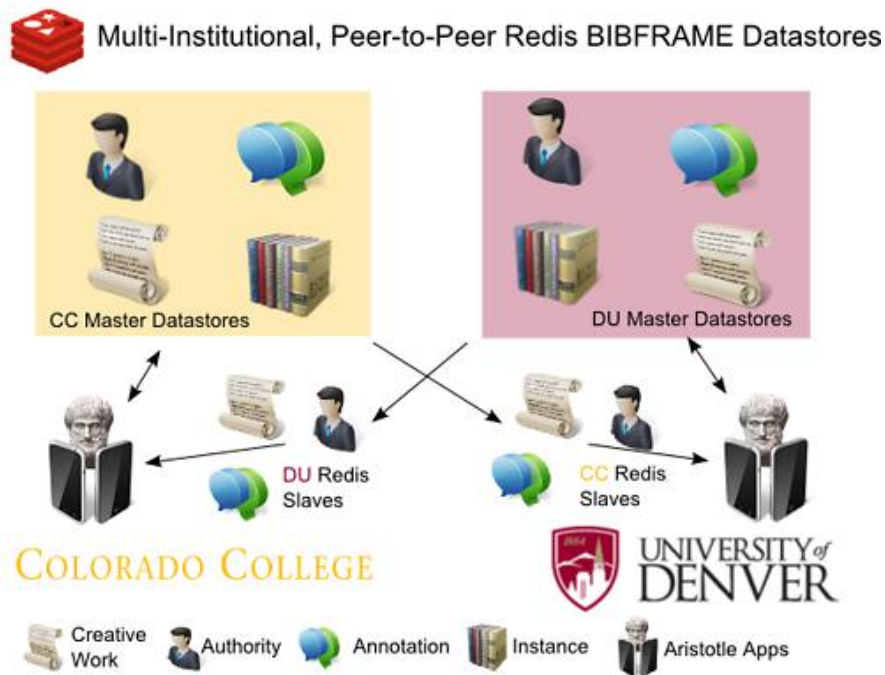


Figure 5. Multiple Institutions Peer-to-Peer BIBFRAME Datastores

We are currently in the early stages of prototyping a peer-to-peer BIBFRAME Datastore with the University of Denver's Penrose Library. As the above diagram illustrates, in this early exploratory work Colorado College and the University of Denver each have their own Redis master running in their respective, local virtual machine environments. Each institution also has local Redis slaves of the other's Creative Work, Authority, and Annotations that sync up with their corresponding master. Our intention is to test the usability of a shared BIBFRAME Redis-based union catalog with two institutions to prepare for scaling up to a consortium-level service.

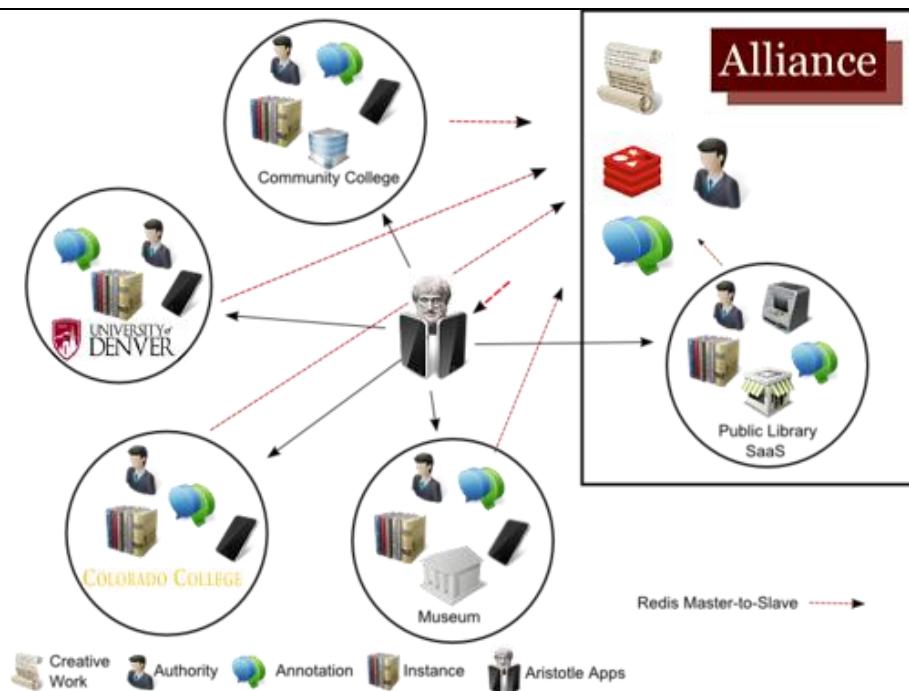


Figure 6. Design of a Consortium BIBFRAME Datastores

Maintaining record-level interoperability should be relatively easy in the Library App Portfolio as long as the MARC utilities and productivity apps are in active development. The challenge is in integrating material requests and real-time circulation status into the proprietary system the Alliance currently uses for the Prospector. A strength of Redis is its ability to store and serve large volumes of data, as it has done for sites such as Github, Engine Yard, Craigslist, Disqus, and Stack Overflow [16]. The library is in early discussions with the Alliance about expanding the Aristotle Library Apps project to scale for millions of records.

Some interesting network topologies for bibliographic information may be possible when using Redis as the underlying datastore and FRBR/RDA as the organizing principle. For example, the Alliance may host the shared Work and Expression datastores, then subscribe to Manifestation and Item datastores managed and hosted locally at each institution, at the Alliance, or at a commercial cloud provider. Each institution could also use the Alliance's hosted Work and Expression datastores in their own Access and Discovery Apps.

To support the increased scale of a consortium or regional BIBFRAME Redis ecosystem, we are closely following Redis Cluster, a sub-project of the Redis open source project. Redis Cluster offers a distributed and fault tolerant [17] environment for very large

and distributed data nodes that matches well with our peer-to-peer and consortium-level BIBFRAME datastores. Redis Custer is under active development, with the developers planning a stable beta release in 2013. The library and the Alliance are also exploring grant opportunities to fund the development and support for a new bibliographic datastore that will scale to hundreds of millions of FRBR entities.

Notes

- [1] pymarc. Available from: <https://github.com/edsu/pymarc>
- [2] Aristotle Discovery Layer Project. Available
from: <https://github.com/jermnelson/Discover-Aristotle>
- [3] Nelson J. NoSQL Bibliographic Records: Implementing a Native FRBR Datastore with
Redis. Code4lib 2012, Seattle, Washington. Available
from: <http://discovery.coloradocollege.edu/code4lib>
- [4] A Bibliographic Framework for the Digital Age. October 31, 2011. Available
from: <http://www.loc.gov/marc/transition/news/framework-103111.html>
- [5] Aristotle Library Apps Project. Available
from: <https://github.com/jermnelson/aristotle-library-apps>
- [6] McCallum, S. Bibliographic Framework Initiative Approach for MARC Data as Linked
Data, September 13, 2012. IGeLU Conference Presentation. Powerpoint available
at: <http://igelu.org/wp-content/uploads/2012/09/IGeLU-sally-McCallum.pptx>
- [7] Miller, E., Ogbuji, U. and K. MacDougall Bibliographic Framework as a Web of Data:
Linked Data Model and Supporting Services. November 11, 2012. Available
from: <http://www.loc.gov/marc/transition/pdf/marclid-report-11-21-2012.pdf>
- [8] BIBFRAME-Datastore project. Available
at: <https://github.com/jermnelson/BIBFRAME-Datastore>
- [9] Redis persistence. Available from: <http://redis.io/topics/persistence>
- [10] Redis Presharding. Available from <http://antirez.com/post/redis-presharding.html>
- [11] ALA's RDA Toolkit Mappings. Available with subscription
at: <http://access.rdatoolkit.org/document.php?id=jscmap1>
- [12] redis-py. Available from: <https://github.com/andymccurdy/redis-py/>
- [13] Marcotte E. Responsive Web Design. A List Apart. May 25, 2010. Available
from: <http://www.alistapart.com/articles/responsive-web-design/>

- [14] Functional Requirements for Bibliographic Records. International Federation of Library Associations and Institutions. December 26, 2007. Available from: http://archive.ifla.org/VII/s13/frbr/frbr_current2.htm
- [15] From the Aristotle Library Apps's Project https://github.com/jermnelson/aristotle-library-apps/blob/master/hours/redis_helpers.py
- [16] Who's using Redis? Available from: <http://redis.io/topics/whos-using-redis>
- [17] Redis cluster Specification (work in progress). Available from <http://redis.io/topics/clster-spec>

用 Redis 和 Django 构建图书馆应用组合

科罗拉多学院的图特图书馆正在开发一个图书馆应用程序组合，供读者和图书馆工作人员使用。这些一次性使用的 HTML5 应用是在迭代和增量的敏捷模式下开发的，针对多种设备，同时使用 Bootstrap 和 Django 向运行在 Redis（一种先进的 NoSQL 数据库服务器）上的底层 FRBR 数据存储提供快速和响应的接口。分为两类：一类是用于访问和发现的应用，人人都可以使用；另一类是生产效率的应用，主要供图书馆工作人员管理 FRBR-RDA 记录。该检索组合包括图书检索、文章检索、馆号检索、馆时检索等应用。效率方面包括一个订单应用程序和一个 MARC 批处理应用程序，用于使用 RDA 核心属性将 MARC 记录摄取为 FRBR 实体。当达到一个关键的阈值时，图特图书馆打算用这个图书馆应用组合来替换其传统的 ILS。

作者：Jeremy Nelson

背景

科罗拉多学院的图特图书馆是一个小型的城市学术图书馆，服务于科罗拉多州科罗拉多斯普林斯的 2000 多名学生和教职员工。作为科罗拉多州研究图书馆联盟（科罗拉多州和怀俄明州的学术和公共图书馆联盟）的成员，我们参加了由成员机构的馆藏组成的联盟目录。我们的 Islandora 机构存储库也是通过科罗拉多联盟的存储库服务托管的。不过，我们使用自己的 III 代的 Millennium ILS 实例。像大多数学术图书馆一样，我们的资料预算已经从实物资源转向电子资源，因此，我们正在为这些电子资源进行更多的 MARC 记录批量加载，而减少了印刷资料的原始或副本编目。

由于供应商提供的 MARC 记录质量参差不齐，旧的工作流程在操作和加载记录到我们的传统 ILS 中往往是漫长而费力的过程。通过使用 Python 和 pymarc Python 模块编写这些 MARC 记录的脚本，并使用 Django 构建的 web 前端，我们为编目人员节省了大量的时间。这产生了第二个 Django 项目，使高年级学生能够自我提交他们的论文或毕业论文以及任何支持数据集，到我们的数字存储库。

在我们研究新发现层面的各种商业和开源选择时，我们也开始了平行的努力。考虑到资金和资源的限制，以及在不同编程语言和环境中维护多个代码库的烦恼，图书

馆决定使用 fork Kochief, 一个基于 Django 的发现项目。这个决定导致了 Aristotle 的发布, 图特图书馆发现层项目在 Apache 2 开源许可下可在 Github 上使用。

2011 年, 我们开始探索使用流行的 NoSQL 技术 Redis 来表示书目和业务信息。这项研究引发了 FRBR-Redis 数据存储项目, 这是 2012 年 Code4Lib 演讲的主题。使用美国国会图书馆书目框架 (BIBFRAME) 倡议的 "数字时代的书目框架 (BIBFRAME)" 作为 FRBR-Redis 数据存储的起点和高级需求, 我们用超过 850 个单元测试来测试使用 Redis 的适用性, 这些测试代表了 MARC21、MODS 和 VRA 元数据, 这些元数据结构为 FRBR 工作-表达-表现-项目实体, 并带有各种 Redis 数据基元。

2012 年 Code4Lib 会议上强调的趋势之一是在图书馆读者的使用上, 还是在开发应用程序的局限性和机会上, 移动设备的重要性日益增加。这促使我们重新思考如何提供书目信息的访问和管理, 从桌面/丰富的网络 "发现" 模式到简化的移动应用程序用户界面, 用于移动和平板电脑本地应用程序, 如苹果的 iOS 和谷歌的 Android 操作系统。我们开始尝试利用 Twitter 的 Bootstrap 对响应式网页设计的支持来构建简化的 HTML5 应用, 这些应用更接近移动和平板应用的设计美学。我们发布的第一个应用是一个书号应用, 它为我们的发现层提供了一个嵌入式小部件, 同时也是一个针对移动和平板设备的独立应用, 但完全可以通过现代桌面网络浏览器使用。这个应用是亚里士多德图书馆应用项目的开端, 可以在网上查看 http://discovery.coloradocollege.edu/apps/call_number/。

自 2011 年以来, 我们继续监测书目框架 (BIBFRAME) 过渡倡议所做的建模工作。随着我们对 Redis 的经验越来越丰富, 我们发现最初实现的 Redis FRBR 数据存储, 用一个 Redis 实例使用多个数据库, 不如让第一组 FRBR 实体 (Work、Expression、Manifestation 和 Items) 中的每一个都作为单独的 Redis 实例在同一服务器上的不同端口上运行灵活。2012 年 9 月, Sally McCallum 在 IGeLU 的演讲中, 首次提供了一个新的书目模型, 她称之为 MARCR, 用于 MARC 资源。这个新模型支持 RDA 和 FRBR, 但核心实体被缩减为四个: 工作、实例、注释和权威。由于 Redis 的灵活性, 我们很快就能使用 RDA 修改现有的 FRBR Redis 数据存储密钥结构, 以遵循这个新模型。2012 年 11 月发表了关于这个新模型的入门文章, 名称改为 BIBFRAME (书目框架), 并对 9 月 McCallum 的介绍进行了小幅修改。即 Work 实体变成了 Creative Work, 模

型的命名空间正式设定为 BIBFRAME（书目框架）。书目框架(BIBFRAME)Redis 数据存储现在是一个独立的开源项目，采用 GNU 通用公共许可证 2 授权，包括文档、每个书目框架(BIBFRAME)实体的 Redis 服务器配置文件和 LUA 服务器端脚本。

关于 Redis 的更多信息

Redis 是一种键值数据存储，是众多新的 NoSQL 数据技术之一，它为数据的表示和使用提供了另一种模式。Redis 以两种方式支持数据持久化：一种是 RDB 模式，定期保存数据集；另一种是 AOF 模式，每次写操作都会保存数据集。虽然两种方式各有优缺点，但大多数图书馆可以采用 RDB 模式与 AOF 模式相结合的方式保存书目记录，保存图书馆事务性数据（如流通统计）。

Redis 与 MARC 记录的平面文件结构和更多传统图书馆系统的关系型数据库有着本质上的区别。Redis 的灵活性允许在一个数据存储中支持不同的信息方案和结构，从而快速开发多个应用程序。Redis 密钥的构造方式允许在其命名结构中嵌入有关数据结构的语义或启发式信息。Redis 假设相关数据使用密钥命名模式，甚至提供了一个全局函数来递增密钥 ID。

在使用 Redis 时，另一个重要的设计考虑因素是与键相关联的数据基元的类型。最简单的值是一个原子字符串。Redis 列表是一个集合基元，它存储无序和重复的字符串值。Redis 集合存储唯一的字符串值，而排序集合则为每个值分配一个排序权重。如果在排序集合中使用权重为 0，Redis 会根据集合中的字符串值进行词法排序。最后一个 Redis 数据基元是一个哈希。一个 Redis 哈希将多个子键与一个 Redis 键关联起来，并通过 HGET 命令返回与该子键关联的值。

Redis 不是关系型数据库，试图复制 RDBMS 是不理想的。在 Redis 中，键是基本结构，而不是像 RDBMS 中的表行。Redis 键也可以作为数据存储中其他键的字符串值，提供了一种粗糙的 SQL JOIN，但在表示键之间的关系时提供了更多的灵活性，而在 RDBMS 中很难或不可能复制。缺点是不同表之间的引用完整性并没有内置在 Redis 中。最终的一致性可以通过应用逻辑或通过涉及 Redis 服务器命令组合的策略来实现。

Redis 字符串、集合、排序集合、列表和哈希数据基元都提供了在 Redis 服务器中

表示库信息的不同方式。Redis 还提供了许多服务器和基元特定的命令，以简化应用开发，包括 EXIST 和 TYPE。对于 EXIST 命令，将一个字符串作为参数传入，并返回一个布尔值，确认该字符串是否是数据存储中的一个键。TYPE 命令在传递键字符串时，返回键所代表的 Redis 数据结构的类型，如果不存在，则返回一个空值。

对于可能无法放入 RAM 中的大型数据集，Redis 项目的首席开发者 Salvatore Sanfilippo 建议进行预分片，将数据存储在不同的 Redis 实例之间进行分解。

亚里士多德图书馆应用和书目框架(BIBFRAME)数据存储项目所使用的命名模式可以很容易地支持用于大型馆藏的预置码，不过就目前而言，一个拥有 4G 内存的虚拟机足以运行科罗拉多学院所有的书目框架(BIBFRAME)数据存储 Redis 实例。下图是我们科罗拉多学院书目框架(BIBFRAME)数据存储的设置情况。

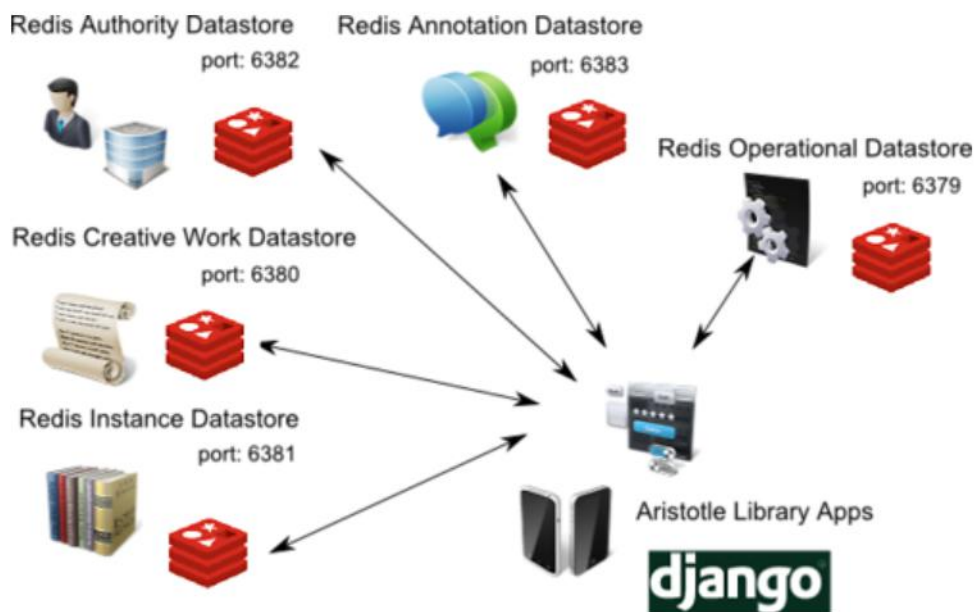


图 1.科罗拉多学院的书目框架(BIBFRAME)数据存储设置

本地书目 Redis 数据存储的演变过程

Redis 书目数据存储的第一次迭代使用了基于 FRBR 和 RDA 的命名模式，使用类似于 XML 命名空间的符号来创建键的集合，以表示值和与数据存储中其他实体的关系。

例如，在亚里士多德图书馆应用项目中，RDA 核心 FRBR 实体和属性是按照 ALA 的 RDA 工具包中提供的 MARC 到 RDA 的映射从 MARC 记录中提取出来的，然后按

照映射到书目框架(BIBFRAME)可链接信息资源(如 Creative Work、Instance、Authority 和 Annotation) 的模式组织成 Redis 键集合。下图显示了一个 CreativeWork 的 RDA 的这样一个书目框架(BIBFRAME) Redis 键集合：包含该实体的支持 Redis 密钥的标题。

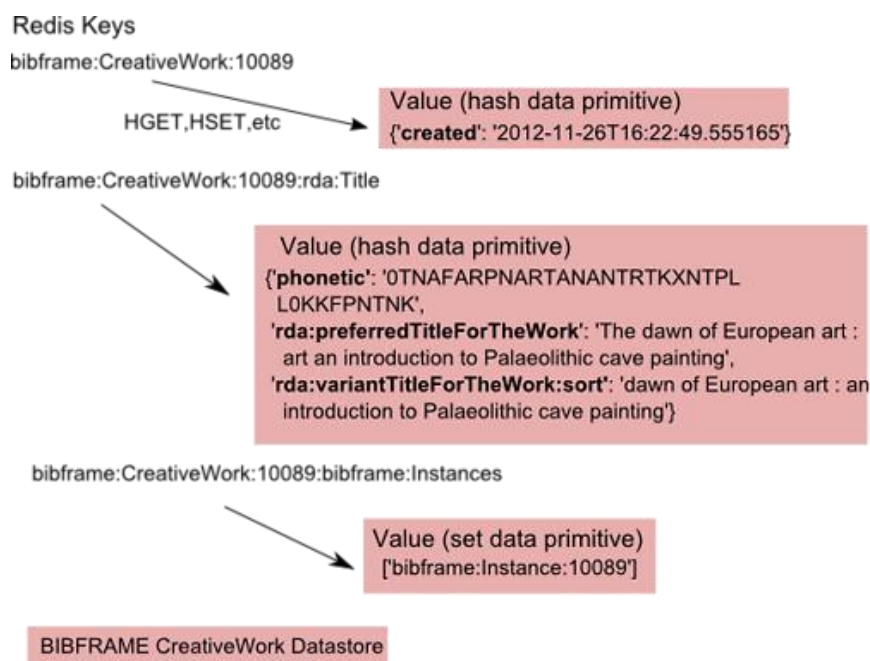


图 2.书目框架(BIBFRAME) CreativeWork 的标题的书目 Redis 键和数据基元值的例子

书目框架(BIBFRAME)数据存储也可以支持其他元数据格式和值。继续前面的例子，如果我们想使用 MODS 或 Dublin Core 而不是 RDA，书目框架(BIBFRAME)数据存储键可以看起来像 `bibframe:CreativeWork:10089:mods:titleInfo`，每个 MODS titleInfo 子元素都是单个 CreativeWork 的哈希键值对。同样，像 `bibframe:CreativeWork:10089:dc:title` 这样的 Redis 键可以是一个简单的静态字符串，也可以是一个 HASH，就像当前书目框架(BIBFRAME)数据存储项目迭代中使用的 `rda:Title` 键一样。我们甚至可以让这三个键(或根据需要的不同键)存在于同一个 Redis 数据存储实例中。Redis 键的结构设计是通过约定俗成和方便性，尽量对 Redis 返回的数据进行明确的描述，而不至于过于啰嗦。

同时，财务信息，如存储在我们 ILS 中的材料订单和发票信息，被提取并添加到 Redis 数据存储中，用于报告和预算预测。我们甚至将图书馆的小时数存储为 Redis 字符串，用于独立的应用程序，并作为 JSON 数据源用于我们的发现层。

Redis 数据存储的交互是通过 Python 类抽象出来的，这些类是用 redis-py 模块构建的。如果应用开发者需要自定义数据存储或扩展 Redis 功能，Python 自定义类可以通过直接操作数据存储来扩展现有的类。

HTML5、响应式网页设计和 Twitter Bootstrap

虽然本地应用程序通常运行速度更快，并且更紧密地遵循各自平台的推荐用户界面指南，但图特图书馆没有资源来维护多个应用程序开发环境。值得庆幸的是，随着网络标准和协议的不断发展，再加上 CSS 和 JavaScript 库的开发，可以创建快速且易于使用的 HTML5 应用，这些应用更具通用性，可以运行在多个移动和平板电脑平台上，以及运行更多支持 HTML5 的现代网络浏览器的个人电脑上。这就是响应式网页设计的目标，正如 A List Apart 的原文所表达的那样：

我们可以将这些设备视为同一体验的不同方面，而不是根据不断增加的网络设备来定制互不关联的设计。我们可以设计出最佳的浏览体验，但将基于标准的技术嵌入到我们的设计中，使它们不仅更加灵活，而且更能适应呈现它们的媒体。总之，我们需要实践响应式网页设计。

亚里士多德图书馆应用程序项目使用流行的网络框架 Bootstrap 作为用户界面的基础，以响应和调整不同的客户端设备和显示。对于一个人员和资源有限的小型图书馆来说，在所有不同的平台、网络浏览器和用户使用的设备上测试应用程序是非常昂贵和不可能的。通过专注于图书馆中最流行和可用的设备（Windows 7，Macintosh，iOS，和一些 Android 手机和平板电脑），图特图书馆针对其读者和工作人员所需的特定功能。这个基于 HTML5 的应用程序开发环境的设计意图是，创建一个新的应用程序的难度应该与建立一个简单的网站大致相当，利用图书馆员和工作人员已有的能力，如 Dreamweaver 和内容管理系统等工具。虽然在对工作人员进行 Bootstrap 和 HTML5 教育时需要进行培训，但与图书馆试图为 iOS 和 Android 环境开发原生应用相比，应用开发的培训负担和要求要小得多。

访问和探索应用程序

在最初的亚里士多德图书馆 App 项目中，大部分的 App 都被归类为访问和探索应用程序，它允许用户查找和访问图书馆的资源以及更多关于图书馆的一般信息。访问和探索应用广泛地解决了用户寻找、识别、选择和获取资源的通用任务，正如 FRBR

规范中所表达的那样。也包括在这个类别中，图特图书馆的时间应用会通知读者图书馆是开放还是关闭。虽然不是书目性质的，但小时应用程序解决了图特图书馆从读者那里收到的最重要的问题之一。

书号应用程序

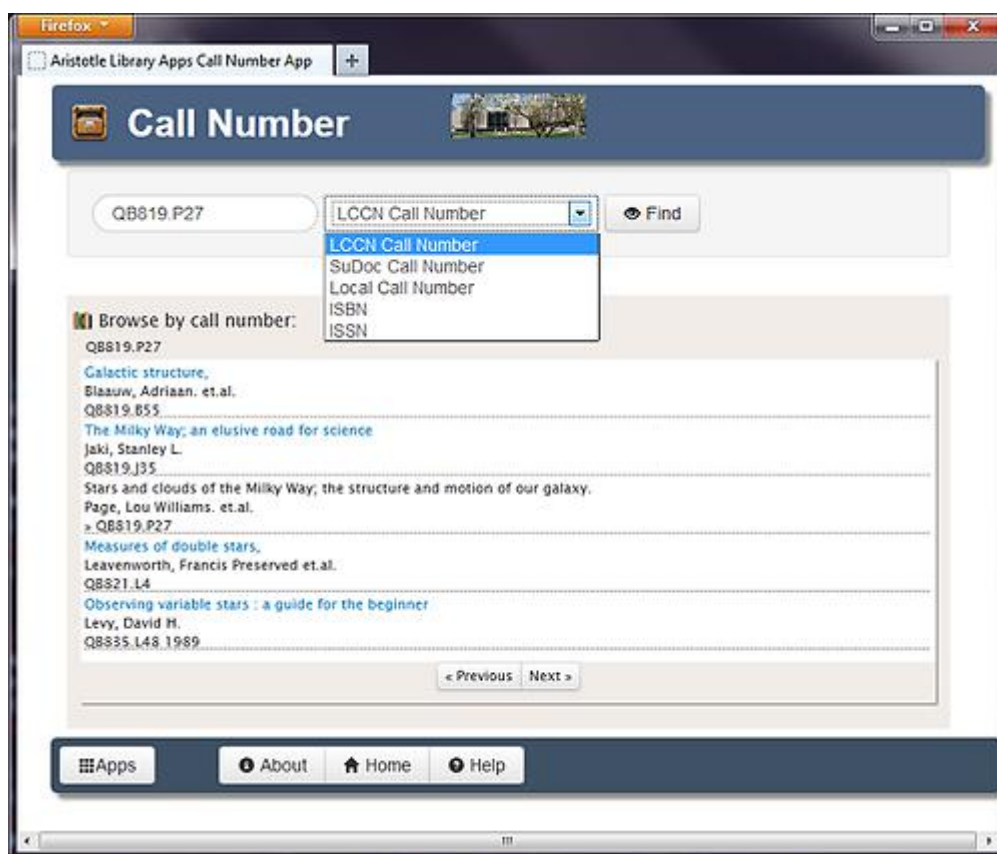


图 3. 书号 App 的截图

书号应用是第一个发布的应用，也是整个 Aristotle 项目的催化剂。在我们研究发现层的时候，另一位图书馆员从斯坦福大学的 Searchworks（用 Blacklight 和 Hydra 构建的）中的一个功能中得到了启发，这个功能允许读者看到图书馆的书库中哪些书号是相互靠近的。在调查斯坦福大学基于 Solr 的实现时，我们意识到可以用 Redis 来简化数据模型。为了创建该应用所需的排序索引类型，将归一化的国会图书馆、SuDoc 和本地书号作为权重添加到 Redis 排序集中。一旦我们将书号应用嵌入到发现层中，我们就探索进一步开发专门的、简化的应用，用于常见的搜索。这些独立的应用程序可以通过使用 JSON API 和原始 HTML，在更大的系统中使用，比如发现层或

图书馆的网站。

图书馆时间应用

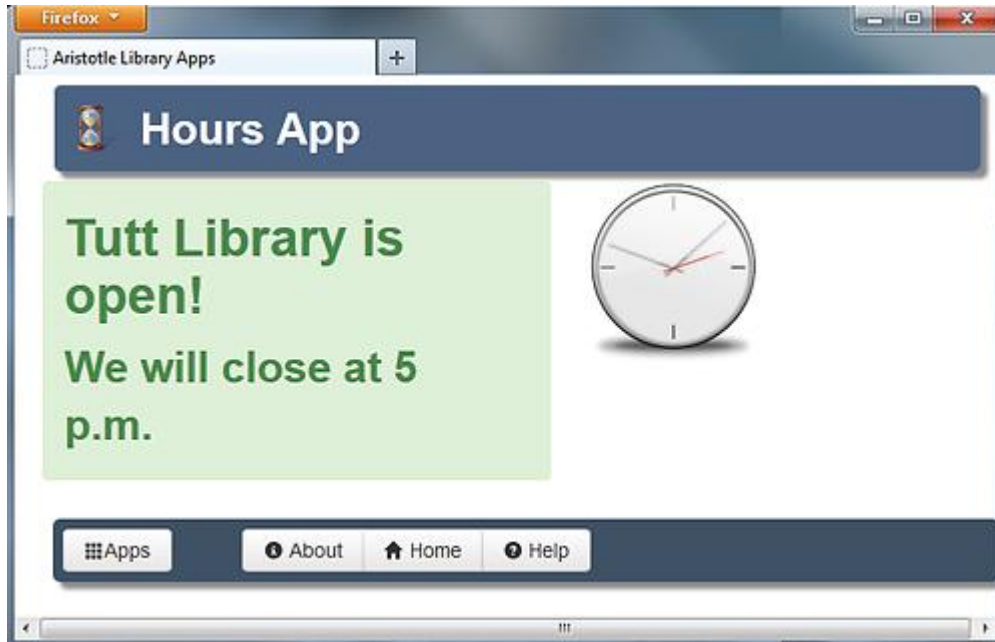


图 4. 图书馆时间应用的截图

当学院采用的 CMS 无法为图书馆主页建立一个动态的图书馆营业时间信息源时，我们觉得用一个专门的 app 来代替 JSON 反馈和存储在 Redis 中的时间数据是可行的。图书馆小时应用程序使用 Redis 位操作命令将日期和小时存储在一个字符串中。每一天都有一个唯一的密钥，其模式如下。"library-hours:YYYY-MM-DD"，其值是一个 96 位的字符串，每一位代表一个刻钟，位偏移量 1 代表 00:00 到 00:14，位偏移量 2 代表 00:15-00:29 等等，代表一天 24 小时中的每一个刻钟。位设置为 0（默认值）表示库是关闭的，位设置为 1 表示库在该刻钟内是开放的。

为了查看库是否打开，时间应用通过使用 Redis GETBIT 命令查询库的运行中的 Redis 数据存储，如时间应用的 redis_helpers.py 中的这段代码所演示的那样：

```
def is_library_open(question_date=datetime.datetime.today(),
                    redis_ds=redis_ds):
    """
    Function checks datastore for open and closing times, returns
    True if library is open.
    <br/>
```

```

:param question_date: Datetime object to check datastore, default
                        is the current datetime stamp.
:param redis_ds: Redis datastore, defaults to module redis_ds
:rtype: Boolean True or False
"""

offset = calculate_offset(question_date)
status_key = question_date.strftime(library_key_format)
return bool(int(redis_ds.getbit(status_key,offset)))

```

时间应用程序的使用者用户界面会显示一条简单的信息，其中包括图书馆当前的时间。如果图书馆关闭，应用程序会在图书馆开放时显示下一个可用的日期和时间。每年 365 个密钥，每天一个，这个 Redis 结构很容易支持时间 App 管理用户界面的要求，允许经过认证的图书馆工作人员添加或修改发布的时间。

生产效率应用程序

第二类应用是针对生产效率的，是为了管理或报告馆藏资源的情况而开发的。这些应用程序需要用户首先进行身份验证，然后根据应用程序和用户的授权，允许对图书馆信息进行操作或报告，其中包括数据存储中的本地书目框架(BIBFRAME)实体。在订单应用程序中，订单记录从图特图书馆的传统 ILS 导入到 FRBR Redis 数据存储中。通过这样做，我们将这些信息从专有的 ILS 供应商那里解放出来，该供应商将订单信息与 MARC 书目记录紧密结合在一起(甚至为订单信息创建自定义的 9xx 字段)。通过将订单信息分离成 Redis 集，将每张发票和订单作为不同的 Redis 键，可视化和预算报告变得更加简单。在没有这个工具之前，我们必须从 ILS 中导出这些数据，并从 MARC21 记录中提取数据，然后进行清理，再导入到 Microsoft Excel 中，对图书馆业务信息这一关键环节进行分析。

我们还开发了一个生产力应用程序，用于图书馆工作人员在数字资源库中移动对象时使用的 Fedora 共享实用程序的集合。摄取一批对象，并对一个或多个 Fedora 对象应用元数据批次更新。虽然这个应用没有直接使用书目框架(BIBFRAME)实体，但它简化了图书馆中使用数字资源库的工作人员的工作流程。亚里士多德图书馆应用项目足够灵活，可以适应其他图书馆系统的工作流程，比如我们的 Fedora 共享数字资源库。

亚里士多德图书馆 App 项目路线图

图特图书馆使用其书号、时间和发现应用程序来增强图书馆的网站和发现层。这些都是公开的，还有文章和书籍搜索应用，网址是<http://discovery.coloradocollege.edu/apps/>。在发现层的记录视图中，同样使用了书号应用程序用来填充书架浏览器的 JSON 接口。时间应用提供了一个嵌入式的 HTML 片段，可以收录在图书馆网站的各个位置。

下一波的应用开发既要关注改善本地工作流程，如资料借阅、课程储备、馆藏清单等图书馆服务，也要关注通过共享书目框架(BIBFRAME)Redis 数据存储改善其他机构资源的可发现性。按照敏捷软件开发的理念，每个应用都应该足够简单，在三到四周的冲刺时间内设计、实现并开始测试，这与科罗拉多学院目前的学术区块日历非常吻合。

点对点 and 联合目录

随着图特图书馆转向应用模式，人们开始关注互操作性，成为任何彻底的技术变革的挑战。科罗拉多州研究图书馆联盟的区域性 Prospector 联盟目录中科罗拉多学院是一个积极的贷款人和借款人，支持学院的街区计划，在那里，学生采取紧张的 3 周半的课程以获得大学学分。由于学生需要及时的研究资料，Tutt 图书馆力争尽快将资料送到学生、教职工手中，最好在 72 小时内送到。基于 Prospector 的 ILL 服务对于满足这种紧迫的材料期限至关重要。任何替代或遗留的 ILS 都不能削弱这种服务。我们应对这些挑战的计划涉及到一种渐进式的方法，先从仅有两个机构共享的书目框架(BIBFRAME)数据库开始，然后再将规模扩大到一个联合体或地区规模。

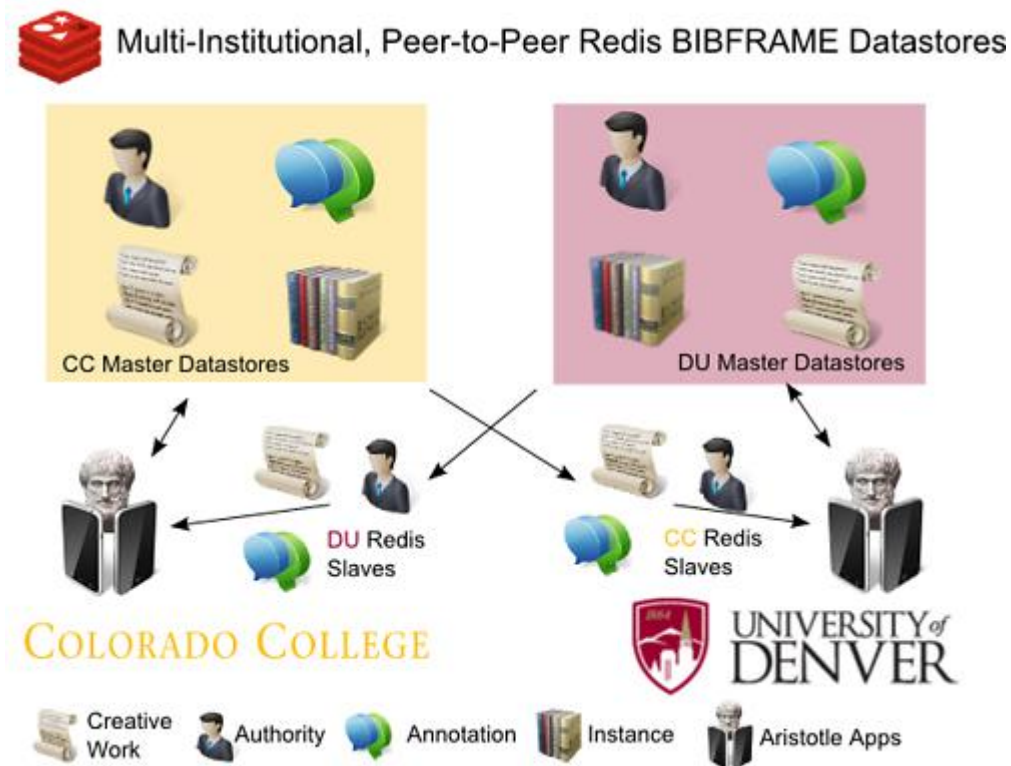


图 5.多机构点对点书目框架(BIBFRAME)数据存储

目前，我们与丹佛大学彭罗斯图书馆合作，正在进行一个点对点的书目框架(BIBFRAME)数据存储的早期原型设计。如上图所示，在这个早期的探索性工作中，科罗拉多学院和丹佛大学在各自的本地虚拟机环境中都有自己的 Redis 主控运行。每个机构也都有当地的 Redis 奴仆，对方的创造性工作、权威性和注释与对应的主人同步。我们的目的是测试与两个机构共享的基于书目框架(BIBFRAME)Redis 的联盟目录的可用性，为扩大到联盟级服务做准备。

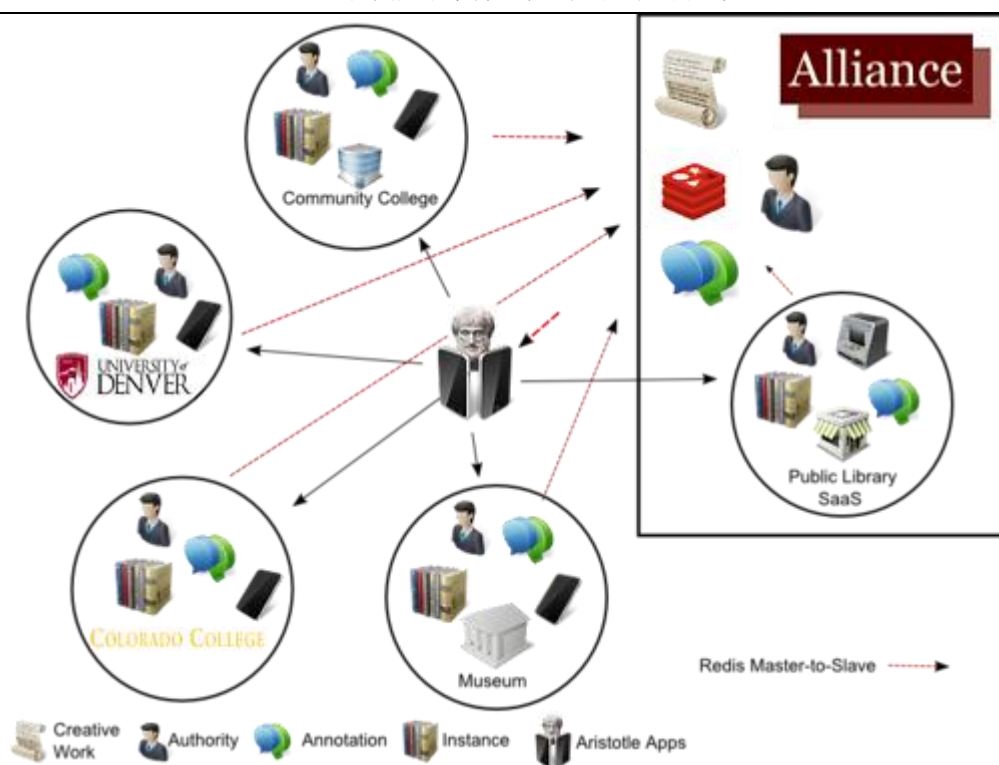


图 6.设计一个书目框架(BIBFRAME)数据存储的联合体

只要 MARC 实用程序和生产力应用程序正在积极开发中，在图书馆应用程序组合中保持记录级的互操作性应该是相对容易的。面临的挑战是如何将材料请求和实时流通状态纳入联盟目前用于 Prospector 的专有系统。Redis 的一个优势是它能够存储和服务大量数据，就像它为 Github、Engine Yard、Craigslist、Disqus 和 Stack Overflow 等网站所做的那样。图书馆正在与联盟就扩大亚里士多德图书馆应用项目的规模，使之达到数百万条记录的规模进行早期讨论。

当使用 Redis 作为底层数据存储和 FRBR/RDA 作为组织原则时，一些有趣的书目信息网络拓扑结构是可能的。例如，联盟可以托管共享的 "工作 "和 "表达 "数据存储，然后订阅由各机构、联盟或商业云提供商本地管理和托管的 "表现 "和 "项目 "数据存储。每个机构也可以在自己的访问和发现应用程序中使用联盟托管的工作和表达式数据存储。

为了支持联盟或区域性书目框架(BIBFRAME)Redis 生态系统规模的扩大，我们正在密切关注 Redis 开源项目的子项目 Redis Cluster。Redis Cluster 为非常大的分布式和分布式数据节点提供了一个分布式和容错的环境，与我们的点对点 and 联盟级书目框架(BIBFRAME)数据存储相匹配。Redis Custer 正在积极开发中，开发者计划在 2013

年推出稳定的测试版。图书馆和联盟还在探索赠款机会，以资助开发和支持一个新的书目数据库，该数据库将扩展到数亿个 FRBR 实体。

参考文献

- [1] pymarc. Available from: <https://github.com/edsu/pymarc>
- [2] Aristotle Discovery Layer Project. Available from: <https://github.com/jermnelson/Discover-Aristotle>
- [3] Nelson J. NoSQL Bibliographic Records: Implementing a Native FRBR Datastore with Redis. Code4lib 2012, Seattle, Washington. Available from: <http://discovery.coloradocollege.edu/code4lib>
- [4] A Bibliographic Framework for the Digital Age. October 31, 2011. Available from: <http://www.loc.gov/marc/transition/news/framework-103111.html>
- [5] Aristotle Library Apps Project. Available from: <https://github.com/jermnelson/aristotle-library-apps>
- [6] McCallum, S. Bibliographic Framework Initiative Approach for MARC Data as Linked Data, September 13, 2012. IGeLU Conference Presentation. Powerpoint available at: <http://igelu.org/wp-content/uploads/2012/09/IGeLU-sally-McCallum.pptx>
- [7] Miller, E., Ogbuji, U. and K. MacDougall Bibliographic Framework as a Web of Data: Linked Data Model and Supporting Services. November 11, 2012. Available from: <http://www.loc.gov/marc/transition/pdf/marclid-report-11-21-2012.pdf>
- [8] BIBFRAME-Datastore project. Available at: <https://github.com/jermnelson/BIBFRAME-Datastore>
- [9] Redis persistence. Available from: <http://redis.io/topics/persistence>
- [10] Redis Presharding. Available from <http://antirez.com/post/redis-presharding.html>
- [11] ALA's RDA Toolkit Mappings. Available with subscription at: <http://access.rdatoolkit.org/document.php?id=jscmap1>
- [12] redis-py. Available from: <https://github.com/andymccurdy/redis-py/>
- [13] Marcotte E. Responsive Web Design. A List Apart. May 25, 2010. Available from: <http://www.alistapart.com/articles/responsive-web-design/>
- [14] Functional Requirements for Bibliographic Records. International Federation of Library Associations and Institutions. December 26, 2007. Available

from: http://archive.ifla.org/VII/s13/frbr/frbr_current2.htm

- [15] From the Aristotle Library Apps's Project https://github.com/jermnelson/aristotle-library-apps/blob/master/hours/redis_helpers.py
- [16] Who's using Redis? Available from: <http://redis.io/topics/whos-using-redis>
- [17] Redis cluster Specification (work in progress). Available from <http://redis.io/topics/cluster-spec>

基于 django 的博客系统开发综述

背景

现如今互联网技术飞速发展，对我们的生活工作学习都产生了非常深远的影响。传统的报纸、书籍等媒介已经越来越无法满足人们日新月异的交流需要，于是，在网上进行交流逐渐成了人们更加偏爱的方式，网络博客就是其中一种交流手段。博客，就是利用互联网，人们可以在上面分享自己的心得、趣事等，也可以与其他人进行实时互动，是内容丰富且个性化的交流平台。博客为人们带来了全新的交流方式、学习方式，甚至改变了人们的生活方式，因此博客受到了原来越多的人的欢迎和喜爱。从具体的定义来说，博客就是一种软件，人们在这个软件上能够发表、传播个人的文章。

如今比较知名的博客网站有如 CSDN：CSDN 博客为中国软件开发者、IT 从业人员、IT 初学者打造交流的专业 IT 技术发表平台,全心致力于帮助开发者通过互联网分享知识,让更多开发者从中受益。LOFTER：LOFTER 是网易 2011 年 8 月下旬推出的一款轻博客，并首次采用独立域名，口号为“专注兴趣，分享创作”。新浪博客：新浪网博客频道是全中国主流，具人气的博客频道。拥有耀眼的娱乐明星博客、知性的名人博客、动人的情感博客，自我的草根博客。

随着科学技术的发展，博客的技术也在随之发展，最开始的博客使用 c 或 c++编写，后来 php 开始流行。最近几年的 Ruby on Rails, Python 也都有用来实现博客系统搭建的博客引擎。^[1]

技术简介

（1）Python

Python 是由 Guido van Rossum 于 1989 年底发明的一种高级程序设计语言，它的设计上坚持清晰划一的风格，由于其易读和易维护的特点，python 成了近几年受到大量欢迎和追捧的语言。Python 是一种简单易读的语言，初学者容易上手，能够使使用者专注于问题本身而不是语言的使用。同时 python 是一种解释型语言，你可以直接从源代码运行而不需要将他转换成二进制代码，这也使 python 具有了更好的可移植

性。另外 python 是一种面向对象的语言，且其从后台系统、系统管理、大数据挖掘、科学计算到 web 开发都拥有丰富的类库和框架。^[2]其中最常见的 web 开发框架有 tomado、web.py、nask、django，目前使用最广泛应用最多的就是 django 框架。

（2）Django 框架

Django 是一个用 web 编写的、开源的应用框架。用 django 开发网站只需要少量的代码就可以开发出一个完整的网站的大部分内容以及全功能的 web 服务。该框架的核心组件有：url 映射机制、模板语言、orm 和表单管理、admin 管理，采用 MVC 模型，即 Model（模型）+ View（视图）+ Controller（控制器）设计模式。MVC 模型简化了对程序的后续修改和开发，并且使程序一部分可以重复利用。使用 django 要先安装 python 和 pip 然后安装 django。python+django 是开发设计一个网站的最好选择，它拥有强大的数据库和后台功能，拥有优雅的网址。^[3]并且 django 拥有很多强大的第三方插件，是他具有较好的可扩展性。

Django 开发的优势在于：（1）功能完备，包括 ORM、视图模板、路由映射、缓存支持等。（2）通用：可以构建各种类型的网站，可以和其他客户端框架一起工作。（3）安全：能够避免许多错误，保护网站。（4）可移植：不受操作系统限制。（5）自动管理后台。

Django 的开发流程包括：部署开发环境——创建项目——创建应用程序——编写业务逻辑代码——建立 url 与视图函数的对应关系——动态加载 HTML 页面——配置静态文件存放位置——连接数据库——django 后台管理。

（3）MVC 模式

MVC 模式是比较流行的一种设计模式，Django 就是按照 MVC 模式开发的框架。MVC 模式就是把 web 分成三层：模型 model、视图 view、控制器 controller。

1 模型是数据存储层，提供数据存储接口，模型从数据库中取数据时，不需要知道数据库取数据的方式。对数据库抽象和封装，因此不需要改代码就可以使用不同的数据库。models.py 创建数据库模型，处理关于存取方式、有效性、数据关系和做由于数据相关的事务。

2 视图是界面，是模型的表现层，用于处理显示什么和怎么显示。Templates 中的模板文件用于显示数据内容。

3 控制器负责业务逻辑，通过逻辑结构决定从数据库中取什么数据，传递什么信息给视图。Urls.py 定义 url 路由表，确定 url 和被调用类之间的关系，通过用户请求调用 views.py 中的函数与数据模型和视图交互，响应用户请求。

另外，Django 的开发模式也被称为 MTV 开发模式，数据模型 model、模板文件 template、视图 view 即 Django 的 web 开发主要集中在 models.py、templates、views.py 中。

（4）数据库 SQLite3

由于本次开发的是一个简单的博客系统，因此我们选择 Django 的内置数据库 SQLite3。SQLite3 是一个轻量级的数据库，只有一个文件，速度很快，方便移动，支持不同操作系统，占用少量资源，利于开发调试。

研究现状

由于国内博客系统的起步比较晚，因此国内对于博客系统的研究也比较晚。国内对于博客系统的研究可以分成几个方向。

（6）对于博客本身的研究。博客的类型，博客的使用等。

（7）对于博客的用户的研究。例如：刘立行、张诗的《视频博客阅听众使用行为研究之初探》^[4]主要探讨了视频博客的用户特征、使用动机和使用情况。雷霄的《浅析现代网络趣缘群体对于媒体平台的使用研究》^[5]主要研究了网络趣缘群体对网易 LOFTER 的使用感受。

（8）博客所涉及的法律问题。例如季渝所写的《论微博作品的著作权保护》^[6]就是对微博客作品的著作权保护问题的探讨。

（9）博客的影响力研究。例如《学术博客影响力评价研究——以科学网博客为例》^[7]（王琛）、《微博客对新闻传播的影响》^[8]（佟文娟、宁健铭）等。

（10）对博客的开发进行的研究，包括基于各种开发框架和技术对博客系统进行构建。例如：《基于 Django 的个人博客系统设计开发》^[9]（常佳宁、李阳齐）主要采用 Django 网页开发框架 Python 语言和 MTV 设计模式实现博客。涂远杰、郑剑的《基于 Flask 的博客网站设计与实现》^[10]主要是利用 Python 中的 Flask 框架开发，同时使用了爬虫技术和 SQLite 数据库，前端使用 HTML、JavaScript 技术。另外李嘉明

基于 Node.js 实现多人博客系统^[1]，和鞠宏军、林涛基于 Spring Boot 和 Hibernate 框架设计和实现博客系统^[12]等等。

研究意义

每个人对于博客都有不同的需求，现在的一个博客网站并不能满足所有人的期望，因此可以创建一个用户自己的博客网站。在保留传统的博客网站的基本功能的基础上，还可以添加一些用户自己期望的功能，增加一些人性化的设计，使用户对博客的操作更加方便。这种可以一个人掌管的实时更新的网站能够提高用户的体验感。

本次设计利用 django 框架完成了一个简易的博客系统的创建，并且分析了 django 的框架结构，体现了 django 在开发时的特点。证明了 Django 和 Python 的技术已经发展成熟，通过 django 框架开发属于自己的博客网站已经完全可行，经过不断改进软硬件方面都已经能够支持。同时能够证明利用 django 开发网站的优越性，可以使复杂的网页开发简单化，能够提高开发者的效率，减少工作量，缩短开发周期。

参考文献

- [1] 邹竞莹.Node.JS 博客系统的设计与实现[D].黑龙江:黑龙江大学,2016.
- [2] 江柳.基于 Django 的博客系统开发研究[J].电脑编程技巧与维护,2016(13):14-15.
- [3] 陈绪.基于 MongoDB 数据库的博客管理系统[D].河北:河北农业大学,2018.
- [4] 刘立行,张诗.视频博客阅听众使用行为研究之初探[J].东南传播,2020,(11):45-49.
- [5] 雷霄.浅析现代网络趣缘群体对于媒体平台的使用研究——以轻博客平台"网易 LOFTER"为例[J].数字通信世界,2018,(11):243-244.
- [6] 季渝.论微博作品的著作权保护[J].法制与社会,2020,(27):33-34.
- [7] 王琛.学术博客影响力评价研究——以科学网博客为例[D].山西:山西财经大学,2018.
- [8] 佟文娟,宁健铭.微博客对新闻传播的影响[J].新闻传播,2017,(7):91,93.
- [9] 常佳宁,李阳齐.基于 Django 的个人博客系统设计开发[J].中国科技信息,2021,(2):75-77.
- [10] 涂远杰,郑剑.基于 Flask 的博客网站设计与实现[J].电脑知识与技

术,2020,16(15):109-111.

[11] 李嘉明. 基于 Node.js 多人博客系统的设计与实现[J]. 电脑知识与技术,2020,16(9):71-72,75.

[12] 鞠宏军,林涛. 基于 Spring Boot 的博客系统的设计与实现[J]. 电脑知识与技术,2019,15(33):50-52,60.

苏州大学本科生毕业设计（论文）中期进展情况检查表

学院（部）：计算机科学与技术学院

学生姓名	黄惠心	年级	2017	专业	计算机科学与技术	填表日期	2021/4/26
设计（论文）选题	基于 Django 的博客系统开发						
已完成的任务	任务书、外文翻译、文献综述已完成。代码完成了大部分内容。正在进行毕业论文。						
	是否符合任务书要求进度	是					
尚须完成的任务	对系统的功能继续完善，完成毕业论文初稿和定稿。						
	能否按期完成任务	是					
存在的问题和解决办法	存在的问题	系统比较基础，需要增加部分功能。文献综述题目不够具体。					
	拟采取的办法	根据指导老师的建议，对发布博文增加文章审查功能，在自主增加部分有意义的功能。修改文献综述题目为《基于 Django 的博客系统开发综述》。					

苏州大学本科生毕业设计（论文）答辩记录表

学院（部）：计算机科学与技术学院

学生姓名	黄惠心	年 级	2017	专 业	计算机科学与技术
设计（论文）题目	基于 Django 的博客系统开发				
答辩时间	2021 年 5 月 14 日	答辩地点	理工楼 239		
<p>答辩小组成员：</p> <p>周夏冰，杨璐，陈文亮，周夏冰，张栋</p>					
<p>答辩中提出的主要问题及学生回答问题的简要情况：</p> <p>问题 1：服务器是怎么实现的？</p> <p>回答：在自己的电脑上运行，安装配置好 django 和代码后，在命令行运行代码，打开浏览器输入网址后就可以运行。</p> <p>问题 2：有没有实际运行展示？</p> <p>回答：有界面截图，中期报告时运行展示过，本次答辩没有带电脑。</p> <p>问题 3：音乐播放器能播放本地还是在线音乐？</p> <p>回答：本地音乐</p> <p>问题 4：有好几个人同时用过吗，可以同时用吗？</p> <p>回答：曾经找同学和老师同时使用测试过，测试成功，可以多人同时使用。</p>					

苏州大学本科生毕业设计（答辩记录表）

答辩小组组长签名：

周夏冰

答辩小组成员签名：

杨璐

薛文亮

周夏冰

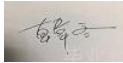
洪林

记录人签名：


洪林

2021 年 5 月 14 日

苏州大学本科毕业设计（成绩评定表）

设计（论文）题目：基于 Django 的博客系统开发			
姓 名	黄惠心	学 号	1727405129
年 级	2017	专 业	计算机科学与技术
指 导 教 师 评 语	<p>黄惠心同学《基于 Django 的博客系统开发》，开发了一款博客的 Web 应用。该博客系统基于 Django 框架，使用 Python 语言进行编程开发；前端使用 Bootstrap 框架，界面简洁美观、操作简单。系统实现了用户注册、登录、发表博客文章、发表评论、文章审查等博客系统基本功能。黄惠心同学对于研究课题能熟练运用本专业所必需的基础理论和专业知识，体现了一定的分析问题，解决问题的能力。论文方案设计较为合理，体现了一定的文献检索能力和阅读及综述能力。该毕业设计也体现了黄惠心同学具备一定的科学素养。在做毕业设计的过程中，黄惠心同学学习态度积极、纪律表现良好。总的来说，该毕业设计（论文）质量较好。综合以上，同意答辩。</p>		
	评 价 内 容		得 分
	设计（论文）方案设计、文献检索、阅读及综述能力、进度等情况评价分（计 25 分）		22
	毕业设计（论文）质量和工作量评价分（计 50 分）		45
	科学素养、学习态度、纪律表现等情况评价分（计 25 分）		23
	成绩（满分 100 分）：90 签名：  2021 年 5 月 13 日		

苏州大学本科毕业设计（成绩评定表）

评 阅 教 师 评 语	<p>本文的工作内容是使用 Django 开发了博客系统。文中介绍了开发本系统所使用的关键性技术以及主要模块。并对系统进行需求分析、系统设计、系统实现和测试。</p> <p>论文选题属于常规选题，基于实际需求，有一定的实用价值。论文书写体现了该生对整体学科知识掌握较好，写作规范也基本符合。</p> <p>修改意见：</p> <p>文中在测试部分只采用单用户模式进行。可以加入并发测试，检验系统的鲁棒性。</p>	
	评 价 内 容	得 分
	毕业设计（论文）文字书写评价分（计 20 分）	17
	毕业设计（论文）质量评价分（计 40 分）	34
	工作量情况评价分（计 20 分）	17
	毕业设计（论文）创新及分析问题、解决问题能力评价分（计 20 分）	15
	成绩(满分 100 分): 83 签名:  2021 年 5 月 13 日	

苏州大学本科生毕业设计（成绩评定表）

答 辩 小 组 评 语	黄惠心同学在毕业设计答辩环节能够运用所学理论和专业知识，简明扼要、重点突出地阐述了《基于 Django 的博客系统开发》论文中相关的主要内容，论点准确，思路较为清晰，语言表达较为流畅。在回答问题环节有理论根据，思路清晰，论点正确，基本概念清楚，对主要问题回答较为正确，有一定创见，表现出理论知识掌握扎实。论文有一定创新性结果，书写基本符合规范化要求。综上所述，该同学所完成的博客系统有一定的实用价值，工作量符合要求。	
	评 价 内 容	得 分
	毕业设计（论文）介绍表达情况评价分（计 20 分）	16
	回答问题表现评价分（计 40 分）	36
	毕业设计（论文）水平和工作量评价分（计 40 分）	34
	成绩（总分 100 分）：86 组长签名：周惠心 2021 年 5 月 14 日	
按权重折算总成绩		86
审定成绩：86		答辩委员会主任签名：朱 2021 年 5 月 14 日

文本复制检测报告单

No: ADBD2021R_2018112714114520210525175744811102382095

检测文献: 基于Django的博客系统开发

作者: 黄惠心

检测范围: 中国学术期刊网络出版总库
中国博士学位论文全文数据库/中国优秀硕士学位论文全文数据库
中国重要会议论文全文数据库
中国重要报纸全文数据库
中国专利全文数据库
图书资源
优先出版文献库
大学生论文联合比对库
互联网资源(包含贴吧等论坛资源)
英文数据库(涵盖期刊、博硕、会议的英文数据以及德国Springer、英国Taylor&Francis 期刊数据库等)
港澳台学术文献库
互联网文档资源
CNKI大成编客-原创作品库

时间范围: 1900-01-01至2021-05-25

检测时间: 2021-05-25 17:57:44

总文字复制比: 6.8%

去除引用: 6.8%

去除本人: 6.8%

重合字数: 1326

文献总字数: 19390

总段落数: [2] **疑似段落数:** [2] **疑似段落最大重合字数:** [717]

前部重合字数: [144] **后部重合字数:** [1182] **疑似段落最小重合字数:** [609]

6.1% 基于Django的博客系统开发.doc 第1部分(总9907字)

7.6% 基于Django的博客系统开发.doc 第2部分(总9483字)