摘 要

目前，社区网站已经非常流行。本文所设计的，就是这样一个人人都可以参与的社区网站，我们可以发表文章，也可以把我们的文档，图片等资料上传到社区来与别人共享。因此，这个社区网站将围绕用户，博客，文件为三个中心，与其相关的若干功能拓展进行设计与实现。

这个系统包含了文字交流分享，以及文件的分享两大块功能，围绕这两个主要功能，又延伸出用户的注册，登录，改密，看帖，发帖，上传，下载文件，留言这些基本功能。我将专注于模型的设计与操作，业务逻辑的实现，而非网页的外观。本文从系统的概述，开发环境，原理分析，模型设计，功能设计与实现全面介绍开发过程。

本文采用MVT设计模式，基于HTTP协议的原理，利用Python语言的Django框架实现业务逻辑，关系型数据库Sqlite3实现数据库层服务，前端开发语言HTML5,CSS,Javascript实现一个简洁的视图层，共同完成了一个文件分享社区系统的设计到实现。

**关键字**：MVT；博客社区；Django；数据库；web。

**Abstract**

At present, the community websites are very common and popular. What I designed, is one such that everyone can participate in comminuty sites, it provides a free service, we can write an article to share with others, also can make our documents, pictures and other data uploaded to the community to share with others.Therefore, this community site will be designed and implemented around users, blogs, files, and related functions.

Share this system contains text, and file sharing two big functions, around the two major functions, and the extended user register, login, change, post, upload, download, comments these basic functions.I will focus on the design and operation of the model, the implementation of the business logic, not the appearance of the web page.This papper I will introduce the system overview, development environment, principle analysis, model design, functional design and implementation.

This article uses the MVT design patterns, based on the principle of the HTTP protocol, implement the business logic using Python Django framework, relational database Sqlite3 database layer service, front development language HTML, CSS, Javascript, realize a simple view layer, completed a common file sharing community system design to implementation.

**Key words**: MVT；Blog community；Django；database；web。

**目 录**

[第一章 绪论 1](#_Toc516742459)

[1.1研究背景 1](#_Toc516742460)

[1.2研究意义 1](#_Toc516742461)

[1.3国内外研究现状分析 2](#_Toc516742462)

[1.3.1 国外情况 2](#_Toc516742463)

[1.3.2 国内情况 3](#_Toc516742464)

[1.4本文的研究内容 3](#_Toc516742465)

[第二章 开发环境及相关技术 4](#_Toc516742466)

[2.1 开发环境 4](#_Toc516742467)

[2.2 使用的技术 4](#_Toc516742468)

[2.2.1 Http协议 4](#_Toc516742469)

[2.2.2 MVT设计模式 5](#_Toc516742470)

[2.2.3 DTL 模板语言 6](#_Toc516742471)

[2.2.4 session 机制 6](#_Toc516742472)

[2.2.5 ORM对象映射方式操作数据库 7](#_Toc516742473)

[2.2.6 Html5，CSS3 ，JavaScript 8](#_Toc516742474)

[2.2.6 非关系型数据库redis 9](#_Toc516742475)

[第三章 系统总体设计 10](#_Toc516742476)

[3.1 方案论证及可行性分析 10](#_Toc516742477)

[3.1.1 方案比较 10](#_Toc516742478)

[3.1.2 结论 11](#_Toc516742479)

[3.2系统模块设计 11](#_Toc516742480)

[3.2.1 总体结构 11](#_Toc516742481)

[3.2.2 用户模块 13](#_Toc516742482)

[3.2.3 博客模块 14](#_Toc516742483)

[3.2.4 文件模块 14](#_Toc516742484)

[3.2.5 站点相关模块 15](#_Toc516742485)

[3.2.6 模板模块 16](#_Toc516742486)

[3.2.7 首页模块 17](#_Toc516742487)

[3.2.8 归档模块 17](#_Toc516742488)

[3.3 系统路由设计 18](#_Toc516742489)

[3.3.1 根目录vshare路由 18](#_Toc516742490)

[3.3.2 aboutsite模块 19](#_Toc516742491)

[3.3.4 其他模块 19](#_Toc516742492)

[3.4 数据库设计 20](#_Toc516742493)

[3.4.1 表设计 20](#_Toc516742494)

[3.4.2 ORM方式创建表 24](#_Toc516742495)

[3.4.3 数据库的增、删、改、查 25](#_Toc516742496)

[第四章 系统软件设计及实现 26](#_Toc516742497)

[4.1 用户模块的实现 26](#_Toc516742498)

[4.1.1 注册 27](#_Toc516742499)

[4.1.2 登录 30](#_Toc516742500)

[4.1.3 个人中心 33](#_Toc516742501)

[4.1.4 改密 35](#_Toc516742502)

[4.1.5 找回密码 36](#_Toc516742503)

[4.1.6 修改密保 36](#_Toc516742504)

[4.1.7 发表博客 36](#_Toc516742505)

[4.1.8 文件中心 37](#_Toc516742506)

[4.2 博客模块的实现 37](#_Toc516742507)

[4.2.1 博客展示 37](#_Toc516742508)

[4.2.2 评论 39](#_Toc516742509)

[4.3 文件模块的实现 39](#_Toc516742510)

[4.3.1 文件上传 40](#_Toc516742511)

[4.3.2 文件下载 42](#_Toc516742512)

[4.4 首页模块的实现 44](#_Toc516742513)

[4.5 归档模块的实现 45](#_Toc516742514)

[4.6 网站相关模块的实现 46](#_Toc516742515)

[4.6.1 网站访问统计 46](#_Toc516742516)

[4.6.2 网站留言板 47](#_Toc516742517)

[4.6.3 关于 49](#_Toc516742518)

[4.7 模板模块的实现 50](#_Toc516742519)

[4.8 其他技术要点的设计实现 50](#_Toc516742520)

[4.8.1 格式化输出时间 50](#_Toc516742521)

[4.8.2 通用分页处理 51](#_Toc516742522)

[4.8.3 验证码的应用 51](#_Toc516742523)

[4.8.4 web 服务器监控 52](#_Toc516742524)

[第五章 系统测试 53](#_Toc516742525)

[5.1 服务器环境配置 53](#_Toc516742526)

[5.1.1 Python环境 53](#_Toc516742527)

[5.1.2 redis数据库 53](#_Toc516742528)

[5.1.3 依赖的Python包 54](#_Toc516742529)

[5.2 项目测试上线 54](#_Toc516742530)

[5.2.1准备项目文件 54](#_Toc516742531)

[5.2.2 修改项目配置 55](#_Toc516742532)

[5.2.3 后台启动web服务 55](#_Toc516742533)

[5.2.4 访问网页 55](#_Toc516742534)

[第六章 总结 56](#_Toc516742535)

[6.1 项目总结 56](#_Toc516742536)

[6.2 项目拓展 57](#_Toc516742537)

[参考文献 58](#_Toc516742538)

[致谢 59](#_Toc516742539)

[附录：项目源码 60](#_Toc516742540)

第一章 绪论

1.1研究背景

网络对于我们来说没有人会感到陌生，20年来我们见证了互联网时代的到来，蓬勃发展，为我们的生活带来了沟通交流，购物，金融，物流，交通乃至方方面面的更加便捷，快速优秀的体验，我们充分享受了互联网科技的成果。

随着20年来的发展，我们先后经历了互联网时代，移动互联网时代，甚至如今已经进入了大数据时代，人工智能时代。不过Web服务架构依然是当前互联网技术的主要应用模式。互联网科技发展到如今，Web服务已经是非常成熟一项业务，功能已经比较完备，但是Web服务需要追求更加高的性能，比如购物网站存在高并发问题解决方案，社交网站需要更好的更人性化，更细致的用户体验，直播平台需要更加稳定流畅的链接和数据传输等性能上的瓶颈。

在这个大的互联网时代背景下，性能是Web服务需要去深度探索，研究的领域。除此之外，Web开发如果能够提升开发效率，那么开发人员就能用更少的精力去创造更大的价值，由此可以花更多的精力去做其他领域研究。

1.2研究意义

我们之所以享受到网络带来的便捷，以Web应用为基础的互联网科技产物功不可没。就是到如今，不论是移动互联网时代，大数据还是人工智能，互联网科技依然是建立在Web应用基础之上的。研究Web开发，能够掌握常用设计模式，理解Web服务的原理以及工作流程。

一方面Web开发这一行业需要探索新的技术，新的领域，来提升性能，提高用户体验，提高应用的效率；另一方面，我们也需要提高Web应用的开发效率，解放开发人员，降低开发成本而发挥Web服务更大的价值。使用Python做Web应用，开发效率很高，可以大大节省人力成本。比如使用Java，php作为Web开发语言的，做一个项目可能需要四个月，而同样的程序员使用Python来做可能只需要两个月。因此，研究Python语言的Web开发，可以极大提高开发效率。Python语言就是这样一门可以大幅提高开发效率的语言。Python作为一款开源语言，拥有非常丰富的第三方库， 这使得开发人员在使用时不需要写很多代码就可以完成很多事情，这是其他语言没法相比的。Python语言的[简洁](https://baike.baidu.com/item/%E7%AE%80%E6%B4%81)性、易读性以及可扩展性，在国外用Python做科学计算的研究机构日益增多，一些知名大学已经采用Python来教授程序设计[课程](https://baike.baidu.com/item/%E8%AF%BE%E7%A8%8B)，Python成为被大量用户所欢迎的、用途广泛的[语言](https://baike.baidu.com/item/%E8%AF%AD%E8%A8%80/2291095)[1]。

此外Python还是一门面向对象的编程语言，它提供了面向函数或者是面向对象的编程方式。面向对象的编程方式具有易维护，易扩展，模块化开发等诸多优点，是开发大型系统的首选方式。通过本次研究，还可以体现面向对象编程的优越性。

1.3国内外研究现状分析

Python在国外的应用十分广泛，从商业到科学研究，大量使用Python，在国内随着这两年人工智能和大数据领域研究的兴起，也有越来越多的公司使用Python作为开发语言，以BAT这样的大公司为代表，也越来越多的研究使用Python。

1.3.1 国外情况

在国外，以美国为代表，做Web开发使用最多的编程语言就是Python。Google公司最主要的语言就是Python，**Google**在很多项目中用python作为网络应用的后端，如Google Groups、Gmail、Google Maps等，Google App Engine支持python作为开发语言[2]。除此之外，比如Reddit - 社交分享网站,Dropbox - 文件分享服务,Fabric - 用于管理大量Linux主机的程序库,Blender -一款开源3D绘图软件，Bittorrent-bt一款下载软件客户端, Ubuntu Software Center - Ubuntu 9.10版本后自带的图形化包管理器,YUM - 用于RPM兼容的Linux系统上的包管理器,Civilization IV - 游戏《文明4》,Battlefield 2 - 游戏《战地2》,NASA - 美国宇航局，从1994年起把python作为主要开发语言,YouTube - 视频分享网站，在某些功能上使用到python[3]。

此外Python在深度学习，机器学习领域也是大放异彩。Google推出的TensorFlow，BVLC推出的 Caffe框架，以及其他一些当前主流人工智能领域的学习框架，绝大部分都支持Python并且将Python作为第一开发语言[4]。

可以发现，在国外Python的使用非常之广泛，社交网站，云盘服务，邮件服务，服务器管理，网络游戏，航空航天，视频网站，图形处理，科学研究领域等皆有Python的身影，可以说是包罗万象。

1.3.2 国内情况

在国内，主要以大公司为首，逐渐有更多的互联网公司开始使用Python。比较知名的有社区网站豆瓣，网络问答社区知乎，阿里巴巴旗下淘宝返利网，做邮件服务的搜狐邮箱，招聘网站拉钩网，租房网站九九房，文化娱乐海报网，这些网站服务器端的主要编程语言是Python。此外，众多小型创业公司最喜欢使用Python，小公司资本不够雄厚，做开发缺乏不了数据，因此需要爬虫程序来获取数据，Python做爬虫代码简洁却功能强大。而开发Web应用则开发效率高，因此不论是做Web开发还是做数据挖掘，Python都是一种越来越受欢迎的语言。

中国的网民数量非常之大，做大数据分析也是具有得天独厚的人口条件，而Python所具有的科学计算库pandas，matplotlib，numpy，都是非常优秀的科学计算，图形处理库，并且使用起来更加方便灵活，国内不少做大数据开发的机构都在使用Python。由百度推出的开源深度学习框架PaddlePaddle首选语言也是Python[5]，并且还推出了一系列的Python深度学习资料，Python在国内的应用范围也逐渐扩大。

1.4本文的研究内容

本文介绍了一个由Python语言的框架Django开发的博客类社区网站系统。通过论述社区的视图层，我将使用Html超文本标记语言，以及控制页面元素样式和行为的CSS，JavaScript对网页和用户交互行为设计到实现[6]；通过论述网站业务逻辑层，我将进行需求分析，方案分析，大量使用Python语言进行面向对象的方式编程实现业务逻辑；通过论述数据库层，我将使用关系型数据库Sqlite3配合需求，设计满足需要的表，使用ORM方式对数据库进行增、删、改、查操作，实现数据库与后台Python的交互过程。最后完成一个完整的具有文件分享功能的博客网站系统。

通过完成这个系统，我将对Django框架的开发流程进行研究，掌握Web服务的工作原理和工作流程，掌握常用设计模式MVC，研究Python语言面向对象的编程模式。

第二章 开发环境及相关技术

2.1 开发环境

本次设计开发环境为Windows10操作系统，Python3.5.2， Pycharm编辑器，Django2.0.1的框架。

2.2 使用的技术

2.2.1 Http协议

HTTP协议是Hyper Text Transfer Protocol（超文本传输协议）的缩写,是用于从万维网（WWW:World Wide Web ）服务器传输超文本到本地浏览器的传送协议[7]。HTTP是一个基于TCP/IP通信协议来传递数据（HTML 文件, 图片文件, 查询结果等）。HTTP是一个属于应用层的面向对象的协议，由于其简捷、快速的方式，适用于分布式超媒体信息系统[8]。它于1990年提出，经过几年的使用与发展，得到不断地完善和扩展。目前在WWW中使用的是HTTP/1.0的第六版，HTTP/1.1的规范化工作正在进行之中，而且HTTP-NG(Next Generation of HTTP)的建议已经提出[9]。HTTP协议工作于客户端-服务端架构为上。浏览器作为HTTP客户端通过URL向HTTP服务端即WEB服务器发送所有请求。Web服务器根据接收到的请求后，向客户端发送响应信息。

url：统一[资源](https://baike.baidu.com/item/%E8%B5%84%E6%BA%90)定位符是对可以从[互联网](https://baike.baidu.com/item/%E4%BA%92%E8%81%94%E7%BD%91)上得到的资源的位置和访问方法的一种简洁的表示，是互联网上标准资源的地址[10]。互联网上的每个文件都有一个唯一的URL，它包含的信息指出文件的位置以及浏览器应该怎么处理它。

url格式：比如https://www.taobao.com 这个url从前往后依次是：协议://主机:端口/路径?参数，协议即https, 这是一种安全认证的http协议标识，而由于使用了DNS服务器，所以主机+端口即为taobao.com，如果有参数则还会加入参数。

DNS解析，即使用特定字符绑定原有的ip+端口的地址表示方式，中间使用dns解析服务器，当一个请求发出时，浏览器的请求会首先发往dns解析服务器，而非真正响应的Web 服务器，dns解析服务器会将浏览器地址中字符进行和ip+端口一一对应的匹配，找到真正需要进行响应的服务器，将这个请求转发给他，而当响应服务器做出响应，浏览器端便显示出渲染后的响应信息。

Http工作流程：浏览器发起请求🡪建立socket套接字🡪dns找到服务器主机，端口🡪建立连接(tcp)🡪服务器返回响应🡪渲染来自服务器的响应数据🡪显示网页🡪关闭连接[11]。

表 2.1 请求方式及其描述

|  |  |
| --- | --- |
| 请求方式 | 描述 |
| GET | 请求指定的页面信息，并返回实体主体。 |
| HEAD | 类似于GET，返回的响应中没有实体内容，用于获取报头。 |
| POST | 向指定资源提交数据进行处理，数据被包含在请求体中。 |
| PUT | 从客户端向服务器传送数据取代指定文档的内容。 |
| DELETE | 请求服务器删除指定页面。 |
| CONNECT | HTTP/1.1协议中预留给能够将连接改为管道方式的代理服务器。 |
| OPTIONS | 允许客户端查看服务器的性能。 |
| TRACE | 回显服务器收到的请求，主要用于测试或诊断。 |

作为一个Web应用，所有的浏览器的行为操作，都是通过请求来进行传递的。而常用请求见于图2.1，此次设计只使用了GET和POST两种请求方式，GET用于普通查询，显示请求，而POST则用于需要传输一些重要数据的请求[12]。此文中a标签，浏览器地址栏这两种情况默认采用GET请求，而用户进行留言，发表博客，登录，注册这些需要向服务器数据库写入数据的则使用form表单构造POST请求，或者使用Ajax发送POST请求。响应则由Django框架中的HttpResponse，HttpResponseRediect，Rediect，Template，render这几个类共同完成，中间的逻辑则需要我自己写。

此处还需要引入状态码的概念，仅作简要陈述，使用时在具体分析。HTTP状态码（HTTP Status Code）是用以表示网页[服务器](https://baike.baidu.com/item/%E6%9C%8D%E5%8A%A1%E5%99%A8)HTTP响应状态的3位数字代码[13]。状态码分为5种，分别是100~500 的整数。1xx表示正在传输消息或等待后续传输，通常在数据分片传输时出现；2xx表示响应成功，正常响应即可；3xx表示重定向，比如在用户登录成功时重定向到指定页面；4xx表示页面或数据丢失，用户浏览器端输入不存在的请求路径或者请求路径移除就会出现；5xx表示服务端逻辑错误，通常因为服务器端接受了请求，但是做出响应时出现逻辑错误造成的。

2.2.2 MVT设计模式

MVC是一种优秀的设计模式，应用十分广泛。他的主要目的是将前端和后台以及数据进行分离，使得开发过程变得更加有条理，易于维护，拓展。

Model（模型）是应用程序中用于处理应用程序数据逻辑的部分。通常模型对象负责在数据库中存取数据。

View（视图）是应用程序中处理数据显示的部分。通常视图是依据模型数据创建的。

Controller（控制器）是应用程序中处理业务逻辑的部分。通常控制器的作用是连接视图层和数据库层，当视图需要数据的时候控制器从数据库读出数据，而需要对数据进行保存时则控制器将来自视图的数据写入数据库。

MVT则是稍加变动的MVC模式，其中V包含了MVC中的VC两部分，新加入T即Templates部分，利用DTL语言进行页面渲染，关于DTL的详细论述，见于2.2.4节。

2.2.3 DTL 模板语言

DTL全称为django templates language，是一种django提供的模板语言[14]。我们所看到的网页是静态的，是浏览器渲染之后的结果。一个网站的内容是变化的，有用户参与的，比如用户发表了一篇新的博客，那么在博客展示页面就会有相应的变动，展示出最新发表的一篇博客，换句话说，后台服务逻辑会从数据库查出这篇文章，把这篇文章传递给页面。由页面来进行渲染展示，完成这个前端与后台之间交互的功能就要使用我们的模板语言Django，下面通过一个简单的例子进行说明。

某个后台逻辑返回一个变量，将他以字典的形式传递给静态页面的html代码，我们需要在html中能够取出这个变量的值，如下的例子：

def function(request):

# 后台逻辑

return render(request, ‘hello.html’, {‘a’: ‘hello’})

则在hello.html文件源码中，我们可以这样使用：

{{ a }}

这样就可以使用字典键’a’ 的值，不仅如此，字典的值还可以是一个对象，一个列表，元组或者其他允许的数据结构，甚至还可以是一个函数，在html中能够使用类似对象访问属性，方法的形式取出后台传过来的值。

2.2.4 session 机制

Http协议是一种无状态协议，上一次请求和下一次请求之间没有关联，所以就算是同一个浏览器，服务器端也不知道上一次浏览器做了什么操作[15]。在一些场景下我们需要知道用户在浏览器端做过什么，比如你在未登录时选择了几件商品，点击结算时需要你前去登录，那么你登录完成之后，服务器端需要知道你之前选择了什么商品，并且你是否已经登录这个状态，服务器也是需要知道但是不能通过请求来知道的，此时就引入了一种浏览器端的存储技术cookie来存储一些关键信息，浏览器下次发送请求时，携带上这些存储在本地的信息，这样就可以辨别浏览器身份，知道浏览器做过什么操作。

首先服务器接收你的请求，当浏览器的操作也就是用户的操作满足一些条件时，服务器给你响应，并且将一部分cookie数据传给浏览器，这些cookie存储在了浏览器端，当某些时候服务器需要使用这些cookie来知道你是谁，你以前做过什么的时候，在检测你发送上来的请求中的cookie信息进行核对。cookie特点，有一个特点，那就是在发送请求时携带cookie信息，但是不会垮域名，百度会在我们的浏览器设置cookie，淘宝也会在我们的浏览器设置cookie，腾讯还是会在我们的浏览器设置cookie，但是向百度发送请求时，只会携带百度的cookie，绝不会将淘宝或者腾讯的cookie携带上去，这就是因为百度，腾讯，淘宝他们有着不同的域名。Cookie的有效时间是可以设置的，比如我们登录之后保存cookie，设置过期时间为关闭会话为止，那么我们在浏览器未关闭之前cookie就是一直生效的，可以完成正常的事情，关闭浏览器之后cookie不再生效，这样也保证了一定的安全性，否则cookie如果一直存在并生效，别人是使用你的电脑就可以随意操作你的个人账号信息，所以关键的信息最好不要直接存储在cookie。通过cookie这样一种本地存储技术弥补了Http协议的不足。

session技术则是在cookie技术的基础上延伸出的更加安全的一种会话跟踪技术[16]。与cookie相比，cookie是将一些关键信息直接存储在了本地，如果有一些不法分子，他们可以直接拿到cookie如果你的这部分数据没有加密或者加密被破解的话，那么就可以直接获取你的私人信息，因此cookie存在很大的安全隐患；session则不同，session依然用到了cookie技术，但是cookie里面存储的并不是关键信息本身，而是关键信息存储的位置，比如redis数据库中的键，这样存储在本地的仅仅是一个信息地址，而不是信息本身，请求时携带这个地址可以在服务器端验明身份，但是拿去这个地址却无法访问地址所指数据库，这样就增加了这种存储技术的安全性。

2.2.5 ORM对象映射方式操作数据库

本次设计采用的数据库Sqlite3是一种轻量级关系型数据库，类似于应用广泛的MySQL数据库，只有轻微的SQL语句区别。这个数据库非常轻量，可以作为一个单独的文件进行使用而无需安装配置环境，这也是我选择使用默认数据库进行存储的原因。

关系型的数据库，可以使用SQL语句进行操作，但实际开发中人们却通常使用编程语言提供的接口来操作，Python语言也提供了执行SQL语句的接口，不过在实际的开发中基本都是使用ORM方式，首先建立模型类，一个类对应着数据库中的一张表，类中的一个属性对应着数据库表中的一个字段，然后通过对象的属性，方法的调用方式来访问修改表的字段，这种方式叫做ORM对象映射方式操作数据库。

2.2.6 Html5，CSS3 ，JavaScript

Web 网页有三要素：元素，布局，交互。HTML语言是一种超文本标记语言，Html5是其中一个应用最为广泛的版本[17]。Html5即是基本元素，网页上能够看到的任何一个像素都是由Html5来进行定义的。Html5的语法是标签，标签通常由<>包裹，并且成对出现，每一个被标签包裹的文本被浏览器进行解析，我们就看到了网页的内容。CSS则是一种辅助语法，用于控制元素的样式，CSS3则是CSS一个版本，用于控制DOM的层叠样式，通俗地讲就是控制每一个元素的位置，如果不加样式控制，则元素会按照文档流的方式默认排列，就会元素依次从上到下，从左往右排列在浏览器界面上，这样是没法给用户使用的。关于本次设计CSS的使用，我使用了CSS的框架Bootstrap，这将会在2.2.8节进行说明。JavaScript则是控制页面与用户交互的，用户在网页上需要进行各式各样的操作，比如点击，发布博客，写留言，有时候还会用到下拉菜单，滑动鼠标滚轮，拖动滚动条这些操作，这就需要用JavaScript脚本来控制这些行为，页面做出对应的变化。

Bootstrap是一个轻量级控制DOM样式的CSS框架，相对于写原生CSS，它提供了更加简便快捷的方式，那就是通过给元素添加特定属性值的方式进行CSS选择器解析，最后达到控制页面样式的效果。具体使用方法参考使用手册即可，此处不做陈述。

jQuery也是本次设计一个主要框架，它用于控制用户操作与浏览器的交互，用户每做一个动作，都有对应的逻辑执行，从而控制页面动态变化。通过jQuery框架操作相比于写原生JavaScript脚本，会更加的方便快捷，并且jQuery中的Ajax模块可用于页面局部刷新时发送异步请求，使用非常方便。

2.2.6 非关系型数据库redis

Redis是一种目前应用广泛的非关系型数据库，它通过键值对的方式进行存储，速度快，支持高并发，在Web应用中常被作为缓存数据库[18]。

本系统就采用了redis数据库作为缓存数据库，使用它进行session的存取，具体使用场景在4.1.2和4.7.3节进行说明。

第三章 系统总体设计

3.1 方案论证及可行性分析

在本次设计中，概括起来由前端，后台，数据库三部分构成。这三个部分中后台和数据库都有多种选择，因此有多种方案可供选择。

3.1.1 方案比较

(1) 方案一

方案：Django框架 + MySQL + Redis

特点：Django框架成熟稳定，可拓展性高，是一套高可用的web框架。MySQL也是一套稳定的数据库系统，使用广泛，符合当今实际网站开发工程。不考虑其他要求的情况下是本次最合适的方案。

成本：实际生产时需要至少一台web服务器，一台MySQL云服务器，一台Redis云服务器可完成一整套的Web服务。在人力成本上Django开发快速高效，可行性高。

(2) 方案二

方案：Flask框架 + MySQL + Redis

特点：与方案一相比，Flask也是Python语言一种优秀的web框架，优点是可拓展性很高，可以很简便的增添，变动功能。缺点是Django框架更加全面，我们只需要在框架中注册admin模块即可快速生成一个简易的网站管理系统，而在Flask中则需要自己设计实现管理系统，实现起来人力成本较高。

成本：生产环境和方案一相当，但是由于需要自定义管理系统，人力成本高于方案一。

(3) 方案三

方案：Django框架 + SQLite3 + Redis

特点：与方案一相比，SQLite3数据库更加的轻量级，使用更加方便灵活，无需安装配置数据库环境，SQLite3数据库可以作为一个单独的文件存在而不依赖其他环境，因此无需配置额外的数据库服务器以及安装数据库服务。但是缺点也很明显，那就是SQLite3不支持远程登录连接，只能在本地和局域网之内操作数据库。综合而论，这个方案还有另一个优点，在使用关系型数据库时可以轻易由SQLite3切换到其他关系型数据库。因此综上所述，这个方案确定为本次设计的方案，成本最低，在需要使用MySQL时可以轻易切换。

成本：由于无需考虑SQLite3数据库的环境，在三个方案中成本最低，实现起来最快。

3.1.2 结论

综合考虑开发效率，稳定性，网站的性能，人力成本这几方面，方案三可以利用很小的成本快速开发出本次这一整套社区系统，后期也可进行拓展，性能优化，是本次设计最佳方案。

3.2系统模块设计

3.2.1 总体结构

如图3.1所示，为本系统的主要架构设计，他们分别对应着项目下面的几个工程文件夹，也是本次设计的主要结构：



图 3.1 系统模块设计

博客系统，那么就有两个主要角色，一是行为的发出者用户，二是用户行为产生的博客，而本次设计还为用户提供了第二个主要行为分享文件，因此就以用户，博客，文件为三个中心，以这三个中心的相关行为，特征进行细化和拓展。

首先，网站需要有一个首页，需要有一些对外公开的信息，浏览器端的用户可以随意访问首页，查看首页以及其他一些次要页面的信息，因此，划分一个首页模块，用于处理浏览器端以游客身份进行访问的页面，也作为整个网站的一个接入口。首页作为网站接入口，则起到了门面以及承接其他各个网页的作用，所以首页首先需要做一个导航栏，提供网站其他模块的入口，让用户鼠标轻点便可以进入到想要前去的功能页面。其次，首页作为门面，应当展示具有网站特点的东西，因此页面主要部分展示用户已经发表博客的缩略信息，缩略信息包含文章的关键信息，比如标题，分类，提供查看博客详情的入口以供用户深入阅读，并提供网友的评论条数或阅读次数展示博客的人气。网站的右侧则提供博客的归档信息，以及博客人气最高，文件的归档信息，文件人气最高的几项，让用户能够快速找到自己需要的东西。首页还应当展示网站logo，网站关键信息，比如公告，网站负责人等这些信息。

用户模块，用户作为博客网站的主要生产角色，首先需要提供游客身份的用户可以使用的功能， 但是不能是全部功能。游客可以使用的部分用以吸引用户使用服务，但是如果使用成本太低，不需要花太多精力竟能完成，那么有可能遭到恶意的攻击，或者发布大量毫无意义的灌水博客，因此需要提供账号，有一些功能必须登录之后方能使用。需要账号，那么必须设计账户的注册，登录，个人中心，密码安全等一些子模块。

博客模块，博客由用户创建，需要对博客进行描述，分类，作者对应，还需要记录博客阅读次数，评论条数，以及各条评论与文章相对应，并且提供用户对博客的操作，比如发表，选择分类，标签，进行评论。

文件模块，做一个文件中心，提供给用户可以上传自己本地的文件，也可以在文件列表点击下载对应的文件到本地，对文件类型加以限制，为保证服务器正常运转也需对文件大小进行限制。文件中心是一个统一的中心，它提供了文件列表，展示文件上传时间，上传作者，下载次数，文件名等关键信息，用户点击其中的文件名可直接发起下载请求。

归档模块，用于对文件，博客进行归档。随着用户的增多，用户发表的增多，用户上传文件的增多，如果不进行归档，那么用户在查找自己需要的博客或者文件时就会十分困难，因此需要设计归档模块来对文件，博客进行分门别类，整理。

站点相关模块，用于对网站的信息进行展示，加强用户与网站的沟通交流。这个系统设计了一块网站的留言板，为游客和会员都提供了留言接口；记录了网站首页被访问的ip地址以及时间，用于监控网站的流量；还用了一个版面展示了博主信息，也就是我的个人信息，也算是为自己做一个广告，推销自己。

模板模块，用于后台和前端交互。后台的逻辑是用来根据前端的用户行为进行响应，要么是返回一个页面进行展示，要么连接数据库查询用户需要的东西，然后返回前端，模板起了非常重要的作用。此次我设计了3个基础模板，3个由基础模板组合的基础模板，3个错误定制模板。所有的页面都是由这9个模板继承，拓展而来，此处不做详细陈述。

3.2.2 用户模块



图 3.2 用户模块

当用户在使用一些特定的功能时，首先检测用户是否已经登录，登录则继续，否则重定向进行登录，登录页面除了账号密码输入框也提供多个选项，尚未注册，和忘记密码，分别是重定向到注册页面，找回密码页面的链接。用户登录完成则判断用户先前发送请求的url，倘若存在则进入目的页面，否则固定跳转到首页。用户登录之后拥有发表文章，发表评论的权限，还有一个安全中心，用以处理用户忘记密码的情况。用户可以进入安全中心修改密码，找回密码，修改密保，提出反馈建议四项操作，其中修改密码需要先验证登录并且验证密保答案，找回密码时需要用户未登录，否则系统判断为已经登录，无法获取找回密码页面，正常进入找回密码页面则需根据提示验证密保进行找回，修改密保则需验证登录并且验证当前密保，反馈则是直接重定向到网站留言板。模块设计如图 3.2 所示。

3.2.3 博客模块



图 3.3 博客模块

博客模块，使用此模块功能，率先验证用户登录状态，未登录则无法使用。提供一个表单用于发表博客，发表时必须给博客一个标题，从待选项中选择一个分类，一个标签，并至少写两个字以上正文，否则会被拦截而无法提交。首页的归档便是对博客按章标签，分类进行归档，而个人中心的“我的博客”即为按用户对博客进行分类。博客成功提交写入数据库，通过用户id字段和用户进行关联，同理通过分类id，标签id可以和对应的分类，标签进行关联。文章发布成功以后，文章详情页会提供评论的表单，并提供已有评论条数以及评论详情。模块设计如图3.3 所示。

3.2.4 文件模块

文件模块的设计，文件由已登录用户在“文件中心”进行上传，限制单个文件大小，以及个人仓库容量大小，文件中心提供上传按钮，点击可选择本地文件进行上传，文件类型符合要求（目前仅仅支持文档和图片），大小（小于20M）符合要求，且个人仓库容量（500M）充足则进行上传，数据库保存上传文件的用户，文件类型，以及文件地址，而文件本身存储在特定文件夹下，以每个用户账号建立文件夹，又以文件类型进行创建文件夹，然后存储对应文件。用户下载时，则从数据库读取文件所在地址，由服务器找到文件以附件的方式传输给浏览器。此外，文件中心还对已上传文件进行了一个列表展示，默认按照下载量由高到低排列，用户可选择按照上传时间由近到远排列，显示文件主要信息，点击文件名即为发送下载请求。模块示意如图3.4 所示。



3.1.5 归档模块图 3.4 文件模块

3.2.5 站点相关模块

本部分设计用于网站整体相关的一些信息。在首页，提供一个名为“建议”的导航按钮，点击跳转进入站点留言板，站点留言板为游客和已登录的会员都提供了留言的表单。已登录用户可以直接填写留言内容即可直接提交，为防止游客恶意灌水，决定控制留言成本。对于游客，需要填写昵称，邮箱，验证码，留言填写完整方可留言，不仅如此，还需后台核对验证码通过。此处设计了一个图片验证码。由后台生成一个五位数随机码，制作为图片传到前端页面，而验证码字符串本身存储在缓存数据库redis中，用户提交后从数据库取出验证码的值与用户表单提交的进行对比，相同则留言成功写入数据库，否则返回错误信息。

第二个网站相关设计为我的个人信息，由一页静态页面组成，是我的一份简历，以及一个个人联系方式二维码。以此来为我自己做广告，推荐自己。

第三个网站相关设计则是首页接到访问请求时，判断是否为第一次访问首页，是则记录下访问ip，访问时间，以此来统计网站流量。

第四，网站的后台管理。本次设计并未设计更加复杂的，自定义的后台管理模块，

相关结构如图3.5 所示。



图 3.5 网站相关模块设计

但是我注册了Django 自带的一个后台管理admin模块，使用时，将需要进行管理的数据库在admin.py当中进行注册即可产生对所注册数据库记录增，删，改三种操作权限，并且不同的表可以任意搭配权限。

3.2.6 模板模块

模板模块，所有的网站页面都由此模块拓展得到。这里我拆分了分别将网站划分为四个部分，通过对四部分进行组合，拓展，又形成了三个基础模板。然后拟定了三个错误定制模板，用以处理出现错误时对浏览器进行响应。

模板即是浏览器端所要渲染的静态页面，在设计时将模板中的一些关键部分替换为变量，在响应的时候将这些变量的值传递过来替换完成后返回给浏览器，这需要用到DTL模板语言。模板结构及作用如图3.6及表3.1所示。

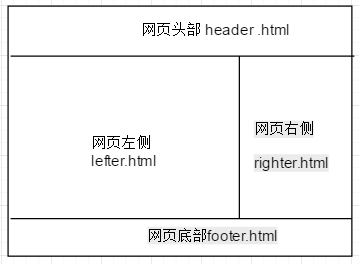


图 3.6 网页拆分示意

表 3.1 模板及其作用

|  |  |  |
| --- | --- | --- |
| 模板名字 | 类型 | 用途 |
| header.html | 网页头部 | 基础模板 |
| lefter.html | 网页左侧 | 基础模板 |
| righter.html | 网页右侧 | 基础模板 |
| footer.html | 网页底部 | 基础模板 |
| base.html | 基准模板 | 基础模板组合构成，拓展生成首页等页面 |
| base2.html | 基准模板 | 基础模板组合而成，拓展生成关于等页面 |
| center\_base.html | 基准模板 | 基础模板组合构成，拓展生成各中心页面 |
| expected.html | 错误定制页面 | 尚未开发的功能模块 |
| 404.html | 错误定制页面 | 未在路由中的请求路径响应 |
| Waring.html | 错误定制页面 | 警示恶意攻击服务器的用户 |

3.2.7 首页模块

首页模块主要是对首页进行展示，提供各个功能的连接，主要涉及html网页制作，而没有太多逻辑，详细说明在4.4节。

3.2.8 归档模块

归档模块，用于对文件，博客进行归档。随着用户的增多，用户发表的增多，用户上传文件的增多，如果不进行归档，那么用户在查找自己需要的博客或者文件时就会十分困难，因此需要设计归档模块来对文件，博客进行分门别类，整理。

归档模块实际上是和文件，博客分不开的，在3.2.3和3.2.4节已经说明结构，具体实现在4.5节。

3.3 系统路由设计

路由是整个Web服务中间部分，作用是根据请求路径确定需要调用的函数或者对象来实现业务逻辑。整个路由分为两部分，因为是模块化开发，因此请求进入项目后首先去根目录urls.py，此文件中的urlpatterns为python语言中的列表，程序会根据列表中的url进行匹配，然后进入子模块路由，子模块路由则再次根据urls.py文件进行匹配，然后执行子模块中相应的逻辑。

3.3.1 根目录vshare路由



图 3.7 系统路由设计

文件在项目中的位置：/vshare/vshare/urls.py

此为项目的主模块，也叫做根模块，当任意一个请求被项目监听到，则请求路径将会背拿到此处urlpatterns列表中进行正则匹配，匹配成功则转入对应的子模块下面的路由继续匹配。跟模块路由如图3.7所示。

服务器端口接收到任意一个请求之后，首先提取请求路径在根目录中进行正则匹配，匹配到则进入对应的模块再次正则匹配下一段url，之后匹配到则调用相应的函数进行响应，否则返回404错误，浏览器上就显示定制的404.html页面。

3.3.2 aboutsite模块

关键代码：

urlpatterns = [

re\_path(r'me\_ljx$', views.MyInfoView.as\_view()),

re\_path(r'msgboard$', views.MsgBoardView.as\_view()),

# 验证码生成和校验

re\_path(r'getcode/\d\*', views.Code.as\_view()),

re\_path(r'checkcode/$', views.CheckCode.as\_view()),

]

功能说明：比如，一个请求路径为”/aboutsite/me\_ljx”，则匹配到urlpatterns中的第一项，执行views.MyInfoView.as\_view() 函数，此函数定义在此模块下的views.py文件中，需要在程序开头导入。其他路由以此类推，依据请求路径匹配情况执行对应逻辑。

3.3.4 其他模块

vshare模块路由中进行include(‘’)，include包裹的便是此系统中所有的子模块，他的原理与3.2.3节基本相同，此处不再陈述，仅列出模块名称：mainpage, archive, article, file, user\_, aboutsite, “ ”,.

上面的 “ ” 表示请求路径为空，即url即ip+端口，没有请求路径部分，此项路由和mainpage路由指向一致，用于直接进入首页第一页， 假如使用”mainpage”这个路由进入首页，则通常会携带一个“num”参数，用于查看首页第几页的博客。

3.4 数据库设计

数据库乃是整个项目最为关键的一部分，根据3.1节需求分析以及方案论证，共设计10张表，由Sqlite3数据库创建，这部分用来存储用户，文章，文件这些需要长期存储的数据。此外，还使用到了Django框架默认生成的后台管理相关12张表，用于管理后台，以及用作缓存的数据库redis一个数据库，存储session和验证码这些临时数据。



图 3.8 E-R实体关系

如图3.8所示，为SQLite3数据库拥有主外键关系的E-R图，由于实体属性较多，此处省略了属性，主键都是各张表自己的id字段，外键依赖就是主键。由于Redis是一种非关系型数据库，不存在表结构以及约束关系，因此不存在E-R图。

3.4.1 表设计

aboutsite模块：共三张表，msgboard用于存储网站留言板提交成功的留言，topnotice表用于存储网站置顶公告，visitdocument表用于记录首页被访问的时间以及访问ip。表结构及其意义如表3.2，3.3，3.4所示。

表 3.2 msgboard

|  |  |  |
| --- | --- | --- |
| 字段 | 数据类型 | 意义 |
| id | 整型 | 主键，确保每条记录的唯一性 |
| create\_time | 时间 | 留言时间 |
| nickname | 字符 | 留言昵称，或会员账号 |
| content | 字符 | 留言内容 |
| email | 字符 | 留言者电子邮箱 |
| is\_delet | 布尔 | 逻辑删除，0为未删除，1为已删除 |

表 3.3 topnotice

|  |  |  |
| --- | --- | --- |
| 字段 | 数据类型 | 意义 |
| id | 整型 | 主键，确保每条记录的唯一性 |
| create\_time | 时间 | 网站公告添加时间 |
| content | 字符 | 网站公告内容 |
| is\_delet | 布尔 | 逻辑删除，0为未删除，1为已删除 |

表 3.4 visitdocument

|  |  |  |
| --- | --- | --- |
| 字段 | 数据类型 | 意义 |
| id | 整型 | 主键，确保每条记录的唯一性 |
| host | 字符 | 访问网页的主机ip |
| visit\_time | 时间 | 访问首页时间 |
| is\_delet | 布尔 | 逻辑删除，0为未删除，1为已删除 |

article模块：共计四张表。tag表用于存储文章标签，category用于存储文章所属分类，article用于存储文章详情，dission用于存储文章的评论。表结构及其意义见于表3.5，3.6，3.7，3.8所示。

表 3.5 tag

|  |  |  |
| --- | --- | --- |
| 字段 | 数据类型 | 意义 |
| id | 整型 | 主键，确保每条记录的唯一性 |
| tag | 字符 | 文章（博客）标签 |
| is\_delet | 布尔 | 逻辑删除，0为未删除，1为已删除 |

表 3.6 category

|  |  |  |
| --- | --- | --- |
| 字段 | 数据类型 | 意义 |
| id | 整型 | 主键，确保每条记录的唯一性 |
| category | 字符 | 文章（博客）分类 |
| is\_delet | 布尔 | 逻辑删除，0为未删除，1为已删除 |

表 3.7 article

|  |  |  |
| --- | --- | --- |
| 字段 | 数据类型 | 意义 |
| id | 整型 | 主键，确保每条记录的唯一性 |
| create\_time | 时间 | 记录文章发表时间 |
| title | 字符 | 文章标题 |
| content | 文本 | 文章内容 |
| read | 整型 | 文章阅览次数 |
| discuss | 整型 | 文章评论条数 |
| author\_id | 外键 | 关联用户id，指明文章作者 |
| category\_id | 外键 | 关联类别id，指明文章所属类别 |
| tag\_id | 外键 | 关联标签id，指明文章标签 |
| is\_delet | 布尔 | 逻辑删除，0为未删除，1为已删除 |

表 3.8 discussion

|  |  |  |
| --- | --- | --- |
| 字段 | 数据类型 | 意义 |
| id | 整型 | 主键，确保每条记录的唯一性 |
| create\_time | 时间 | 评论发表时间 |
| content | 字符 | 评论内容 |
| article\_id | 外键 | 关联文章id，指明这是哪篇文章的评论 |
| user\_id | 外键 | 关联用户id，指明评论的主人 |
| is\_delet | 布尔 | 逻辑删除，0为未删除，1为已删除 |

file模块：共计两张表。filetype用于存储文件类型，目前只支持文档和图片两类。file用于存储文件信息。具体表结构及其意义见于表3.9，3.10所示。

表 3.10 filetype

|  |  |  |
| --- | --- | --- |
| 字段 | 数据类型 | 意义 |
| id | 整型 | 主键，确保每条记录的唯一性 |
| filetype | 字符 | 文件类型，img或者doc |
| name | 字符 | 文件类型名，文档或图片 |
| is\_delet | 布尔 | 逻辑删除，0为未删除，1为已删除 |

表 3.9 file

|  |  |  |
| --- | --- | --- |
| 字段 | 数据类型 | 意义 |
| id | 整型 | 主键，确保每条记录的唯一性 |
| name | 字符 | 文件名 |
| path | 字符 | 文件上传后在服务器上的位置 |
| content\_type | 字符 | 文件格式 |
| upload\_time | 时间 | 上传时间 |
| download | 整型 | 下载次数 |
| filetype\_id | 外键 | 关联文件分类id，说明文件分类 |
| user\_id | 外键 | 关联用户id，说明文件上传者 |
| is\_delet | 布尔 | 逻辑删除，0为未删除，1为已删除 |

user\_模块：一张表，存储用户的账号信息。具体表结构及其意义见于表3.11所示。

表 3.11 user

|  |  |  |
| --- | --- | --- |
| 字段 | 数据类型 | 意义 |
| id | 整型 | 主键，确保每条记录的唯一性 |
| account | 字符 | 用户账号 |
| pwd | 字符 | 用户密码 |
| question | 字符 | 用户密保问题 |
| answer | 字符 | 用户密保答案 |
| phone | 字符 | 用户电话，可以为空 |
| tip | 字符 | 签名，可以为空 |
| modified | 时间 | 变更时间，修改一条记录的任一项都会刷新这个时间 |
| is\_delet | 布尔 | 逻辑删除，0为未删除，1为已删除 |

3.4.2 ORM方式创建表

就以表3.11为例演示使用ORM创建表的过程。

第一步：编写模型类。位置为user\_模块下的models.py文件，类中每一个类属性就对应表的一个字段，

from django.db import modelsclass User(models.Model):  
 id = models.IntegerField(primary\_key=True, auto\_created=True)  
 account = models.CharField(max\_length=20, unique=True,verbose\_name='账号')  
 pwd = models.CharField(max\_length=20, verbose\_name='密码')  
 question = models.CharField(max\_length=40, verbose\_name='密保问题')  
 answer = models.CharField(max\_length=20,verbose\_name='密保答案')  
 phone = models.CharField(max\_length=11, verbose\_name='电话')  
 tip = models.CharField(max\_length=40, verbose\_name='签名')  
 modified = models.CharField(max\_length=19, verbose\_name='变更时间')  
 is\_delet = models.IntegerField(verbose\_name='是否删除',default=0)  
  
 class Meta:  
 verbose\_name\_plural = '账户'  
 ordering = ['id']  
  
 def \_\_str\_\_(self):  
 info = '%s,%s,%s,%s,%s,%s,%s,%s'%(self.account,self.pwd,self.question, self.answer, self.phone, self.tip, self.modified, self.is\_delet)  
 return info

其中verbose\_name\_plural 这个变量的值指定了表在格式化输出时的名字，在进行后台管理时会更加明确表的作用，\_\_str\_\_()方法则指定了格式化输出时输出的表的字段，一般表的字段全部输出。Ordering 变量则表示按照id字段进行从小到大显示表中的每一条记录。

第二步：创建迁移文件。这一步的原理是利用Django内置的模块将上面编写的python语言的模型类代码转换成对数据库操作的SQL语句。操作为命令终端进入vshare根目录，执行命令”python manage.py makemigrations user\_”，其中”user\_”为所在模块。

第三步：执行迁移文件。执行迁移文件就是执行SQL语句，生成对应的表。操作为终端执行命令”python manage.py migrate”。

其他模块中的模型类依照上面三步即可生成对应数据库表，注意将模块名替换为自己的模块名。

3.4.3 数据库的增、删、改、查

以上一节的User表为例，此处简要演示数据库的增、删、改、查操作，更加复杂的操作也是类似的用法，具体应用参考各个模块的实现。

（1）增加一条记录

Django 的model模型提供的create方法，传入每个字段的值即可。

User.objects.create(

account = ‘lujianxin’,

pwd=’123456’

)

（2）删除一条记录

delet方法，传入过滤条件，比如此处删除id字段值为1的那一条记录。

User.objects.delete(id=1)

（3）查询一条记录

查询id为2的那一条记录。

User.objects.get(id=2)

（4）修改记录

修改记录的前提是需要查询出你将要修改的那条记录，比如（3）中查询到的记录，修改他的密码为“654321”.

user = User.objects.get(id=2).update(pwd=”654321”)

（5）使用查询结果

要使用查询到的结果，比如（4）中查询到的user记录，我们想使用他的account字段的值，只需要写如下代码：

account = user.account

第四章 系统软件设计及实现

首先，创建Django项目。方法是，打开Pycharm编辑器，新建Django项目，命名为vshare，同时还要选择编译环境，如果电脑上同时安装了多个python版本，则要手动选择。打开项目可以看见里面有vshare文件夹，有文件manage.py文件，wsgi.py等文件，其中vshare文件夹与项目同名，即为根模块。

其次，为各模块准备基础模板。在刚创建的项目下有一个“templates”文件夹，是Django项目专门用于存放模板文件的目录，这个文件夹名字也可以自定义，但是需要在配置文件中进行配置。在templates目录中建立表3.1中的基本模板，具体代码见于附录部分项目源码。进入settings.py修改其中的配置，如图4.1，4.2所示

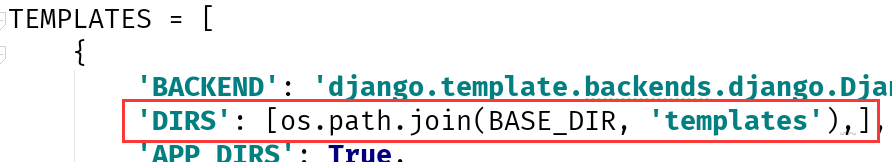


图 4.1 settings配置templates目录



图4.2 templates 配置

然后，进行子模块的开发，在接下来的4.1,4.2,4.3,4.4,4.5,4.6节分别进行说明。

4.1 用户模块的实现

项目终端Terminal执行命令“python manage.py startapp user\_”，创建user\_模块，由于Django框架自带后台管理系统中user常与自定义模块命名冲突，因此故带上后下划线加以区别。创建完user\_模块，打开根模块下的urls.py，在其中添加路由：

re\_path('^user\_/', include('user\_.urls')),

然后去根模块settings.py中找到INSTALLED\_APPS列表，在列表中添加一个字符串：”user\_”，注意与前面的字符用逗号分隔开，否则运行项目会报错。此配置表示注册刚才生成的user\_模块，否则模块无法被识别。

4.1.1 注册

用户访问网站而未登录时，在首页导航栏会出现“登录/注册”连接，点击默认进入登录页面，登录框的左下角有一个“尚未注册”按钮，点击可进入注册页面，注册页面地址为”/user\_/regist”，直接在地址栏输入地址回车同样可以访问注册页面，注册页面如图4.3所示。



图 4.3 注册页面

访问这个页面发送的是GET请求，点击“确认注册”发送的是POST请求。实现这些功能的步骤如下：

（1）编写模板文件

模板文件继承于’base.html’，存放于user\_模块下的templates文件夹，使用Html5,css,JavaScript编写静态页面，此处不便大量粘贴代码,仅展示关键的表单部分。

<form action="/user\_/regist" method="post">……</form>

请求路径为”/user\_/regist”，请求方式为post，因此在Regist类中重写post函数来处理这部分表单提交后的逻辑，流程见于图4.4 所示。



图4.4 注册流程

（2）编写路由文件

注册时，用户在页面中点击“注册”按钮，页面对应着“/user\_/regist”的请求路径，我们在django项目vshare中的urls.py中定义有这样一段路由代码：

from django.contrib import admin

from django.urls import re\_path, include

from django.conf import settings

urlpatterns = [# 各模块的路由包含到此

re\_path('^user\_/', include('user\_.urls')),

re\_path('^article/', include('article.urls')),

re\_path('^aboutsite/', include('aboutsite.urls')),

re\_path('^file/', include('file.urls')),

]

当我们请求路径为“/user\_/regist”的时候，会以正则表达式的形式匹配到re\_path('^user\_/', include('user\_.urls'))这一项，这一项的前面那个参数就是正则表达式，和他相匹配就会执行第二个参数，也就是将user\_模块下的路由文件给包含进来，user\_下面的urls.py关键代码如下：

re\_path(r'regist$', views.Regist.as\_view()),

请求路径会和这句代码相匹配，执行views.Regist.as\_view()函数，这个函数定义在与此处的urls.py文件同级目录下的views.py文件中，django会自动提取请求方式执行其中的与请求方式同名的函数。

（3）编写视图

视图位于views.py中，当接收到的请求路径为“/user\_/regist”时，系统自动判断请求方式，假如请求方式为get，就执行get函数，为post就执行post函数，这个Regist是一个视图类，他继承自django的View类，我们要做的就是写我们自己需要的响应函数，此处我写了get，post两个函数分别用来响应两种方式的请求。响应函数名字必须是表2.1中的请求方式名称小写，他接收两个参数，一个是面向对象编程方式中的关键字self，另一个则是请求信息request。在提取请求参数的时候，比如：

account = request.POST.get('account', '')

依据这个方式就可以提取用户在页面中填写的name属性为account的文本，这就是提取post方式中参数的方法，同理其他方式请求只需更改POST为对应方式也可提取各自请求携带的参数。提取参数之后，则进行参数的合法性判断，下面以\_is\_account\_valid()函数为例说明对于数据合法性的判断。

# 账号格式校验

def \_is\_account\_valid(self, account):

import re

result = re.match(r'^[a-zA-Z\_][a-zA-Z0-9\_]{1,19}$', account)

if result:

return True

else:

return False

在上面这段代码中，通过python标准库中re模块进行正则匹配，re.match()的第一个参数“^[a-zA-Z\_][a-zA-Z0-9\_ ]{1,19}$”表示account这个文本必须以字母下划线开头，整体由数字字母下划线组合，长度为2-20位，满足这些条件则返回真，否则返回假。其他合法性校验与此类似，都是利用正则表达式对用户输入的信息进行限制。

在Regist类中，我写了很多个校验表单数据合法性的方法，他们的名字及作用如表4.1所示。

表 4.1 注册数据合法性校验

|  |  |
| --- | --- |
| 方法 | 校验作用 |
| \_is\_account\_valid | 限制账号为数字字母下划线组合，长度2~20，数字不能开头 |
| \_is\_pwd\_valid | 限制密码数字字母下划线组合，长度6~20 |
| \_is\_sure\_pwd | 限制确认密码和密码相同 |
| \_is\_answer\_valid | 限制问题长度2~40 |
| \_is\_question\_valid | 限制答案长度2~20 |
| \_is\_phone\_valid | 限制电话号码是一个合法号码 |
| \_is\_account\_exited | 校验账号是否已经存在 |

我们编写这些东西就可以完成对于注册功能的实现，其他模块以及功能都是如此的一个过程，主要就是模板，路由，视图三部分不一样，可参考此节。数据库部分已经在3.4节详细说明。

4.1.2 登录

登录页面较为简单，最终效果如图4.5所示。



图 4.5 登录页面

其中对于请求方式的处理与注册相同，点击确认登录之后，后台提取提交的账号和密码，查询数据库中user\_表的所有记录，查看是否有记录is\_delet字段的值为0，并且账号和密码与用户提交能够完全匹配的记录，存在则表明登录成功，登录成功则获取refer\_url，没有获取到或者获取到的url仍然为regist或login则重定向到主页，否则重定向到refer\_url。成功登录后设置session，session内容为登录成功的user对象。Session的配置同样在settinps.py中完成，配置内容及其作用如图4.6所示。



图 4.6 redis配置

登录流程如图4.7所示。



图 4.7 登录流程

用户在浏览器地址栏输入“/user\_/login”或者点击带有这个连接的a标签都会发起登录的get请求，之后参考4.1.1节注册流程，会对这一请求进行响应。其中存储session时：

request.session['account'] = account

这样一种类似于Python中字典的存取方式来进行Redis数据库的session存储。此处为了检测浏览器端的用户登录状态，还定义了一个登录验证中间件login\_check\_middleware.py文件，以及一个用于记录已登录账号的全局上下文logined\_user\_prossor.py文件，这两部分都需要在settings.py中进行注册。中间件和全局上下文的原理是，项目接到任意一个请求，首先执行这个中间件。每次访问都会从session中尝试提取user对象，如果提取到了则将读取到的user对象设定为全局上下文，user对象将会被携带给所有的页面模板，然后正常进行请求的路由，否则提示登录不处理请求而是直接重定向到登录页面。如图4.8所示。



图 4.8 登录验证中间件

中间件验证未登录的提示如图4.9所示，确认之后重定向到登录页面。全局上下文的定义，关键代码如下：

class LoginCheck(object):

redirect\_str = '''

<script>

alert("检测到您并未登录，此功能需要登录后方可使用，请前去登录！");

window.location.href = '/user\_/login';

</script>

'''

# 保存请求内容

def \_\_init\_\_(self, get\_response):

self.get\_response = get\_response

# 重写\_\_call\_\_,对象被创建时执行下面的逻辑

def \_\_call\_\_(self, request, \*args, \*\*kwargs):

# print(request.path)

if request.path in LOGIN\_AUTH\_LIST:

if not request.session.get('user'):

return HttpResponse(LoginCheck.redirect\_str)

# 验证通过则执行刚才的请求，该干啥干啥

return self.get\_response(request, \*args, \*\*kwargs)

此处定义了一个LoginCheck类，当他创建实例时，会调用\_\_call\_\_方法，去session中提取“user”这个对象，提取到则正常执行之前的请求，否则提示未登录。

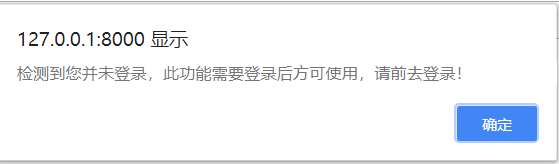


图 4.9 中间件提示

4.1.3 个人中心

在首页导航栏存在“个人中心”按钮，点击可以进入个人中心，但是需要登录之后方可访问，否则会进行4.1.2 节中间件重定向，强制要求登录方可使用此部分功能。

成功进入个人中心，显示欢迎账号的信息，并显示当前系统时间，以及根据时间问候账号。下方栏目显示帐号基本信息，提供了账号的改密，密保，签名等账号信息的操作入口，提示上次更改个人信息的时间。再下方则提示当前账号在社区发表的博客以及上传的文件，点击“查看”按钮可以看到详情，详情页面提供删除已发表博客的入口，以及修改已发表文章入口，暂不支持删除已上传文件。最下方则展示当前账号在社区最近发表的10篇博客和10个最近上传的文件，点击则可阅读博客或下载文件。右侧部分则是网站整体的归档信息，由mainpage模块定义全局上下文实现。个人中心页面如图 4.10所示。



图 4.10 个人中心

个人中心主要逻辑是根据登录用户的id去数据库查询与此用户相关的信息，然后通过模板语言字典传递参数的方式给个人中心的模板，渲染之后便可得到图4.10页面。提供的入口以及实现方法如表4.2所示。

表 4.2 个人中心入口

|  |  |  |
| --- | --- | --- |
| 按钮 | 请求地址 | 后台逻辑 |
| 修改签名 | /user\_/addtip | 更新user表tip字段，覆盖于原来的值 |
| 密码修改 | /user\_/cpwd | 验证密保，通过则更新user表pwd字段，注销当前用户 |
| 密码找回 | /user\_/find\_pwd | 同密码修改，但是无需登录 |
| 社区贡献 | /user\_/usercenter | 通过参数act来确定查询当前账号的所有博客或文件 |
| 最近动态 | /user\_/usercenter | 通过参数r\_files,r\_arts查询当前账号最近的10条记录 |

4.1.4 改密

改密这一功能在网站页面上有两个入口，其一是4.1.3节所介绍的个人中心，进入个人中心后有一个“更改密码”连接，其二是首页顶部导航条“安全中心”连接，进入可以查看与账号有关的操作，其中就有修改密码。改密的逻辑很简单，如图 4.11 所示。



图 4.11 改密

登录部分依然是三部分，前端做什么，后台做什么，数据库做什么，参考4.1.1节实现过程，后台逻辑位于user\_/views.py文件，前端文件在后台文件同级目录templates下的usercenter.html，cpwd.html等文件。目前的设计还比较粗糙，验证密保问题的答案作为修改密码以及找回密码的唯一凭据，道理就是回答对了密保问题，就允许你修改密码，找回密码，更新数据库中的记录，反之就不提供修改密码的页面，不进行修改数据库。是否登录则由4.1.2节中的登录验证中间件完成。

4.1.5 找回密码

找回密码，暂时做的比较简单，就以密保问题答案作为唯一标准，核对密保答案通过则允许设置一个新的密码，否则用户只能通过联系管理员来进行人工干预找回密码。逻辑较为简单，此处不进行详细说明，参考4.1.4节。

4.1.6 修改密保

修改密保和修改密码流程大致相同，参考图 4.11 ，此处不进行详细说明。

4.1.7 发表博客

首页导航条中有一个“写博客”连接，点击之后首先验证登录状态，通过则进入博客撰写页面，否则重定向到登录页面，登录完成可继续写博客的操作。另一个入口为个人中心导航条“发表博客”按钮，同样验证登录状态，请求地址均为“/user\_/write\_article”。后台接收到”/user\_/write\_article”请求，请求方式为get则认为写博客的请求，服务器返回博客撰写页面，请求方式为post则认为是用户提交写好的博客要进行发表，之后开始验证表单数据的合法性，全部通过后进行数据库写入操作。此过程将在博客模块中进行详细介绍。

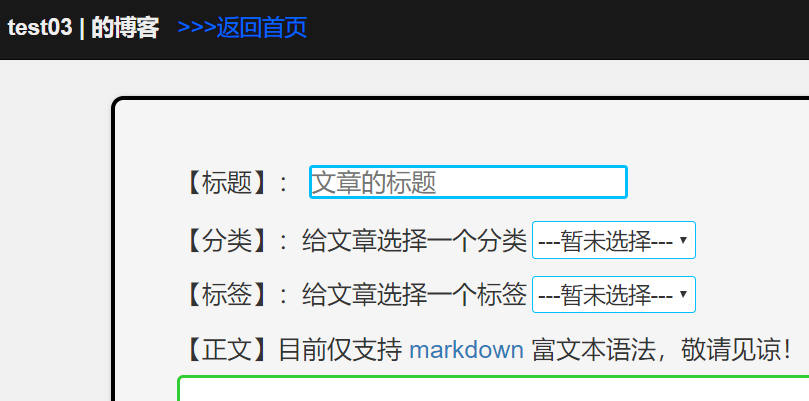


图 4.12 博客发表

博客撰写的文本编辑器暂时较为简陋，仅支持markdown语法，而无法插入图片表情，以及自由编辑段落字体样式。必须为博客填写标题，选择一个分类和一个标签否则无法提交。详细逻辑实现请见4.2.1节。

4.1.8 文件中心

首页导航条“文件中心”，可查看所有用户上传的文件，默认按照下载量由高到低分页显示，标题栏按钮“按时间”，“按下载量”点击可切换排序规则。列表显示文件下载量，上传帐户名，时间等关键信息，页面顶部提供上传入口，也是本站唯一的文件上传入口，文件上传的具体细节请见4.3节，文件中心页面如图4.13 所示。



图 4.13 文件中心页面

4.2 博客模块的实现

博客模块article，结构类似于user\_模块，也需创建新的app，并在settings.py中进行配置。博客发表实际上已经在4.1.7节有所说明，之所以如此设计，是因为博客是由用户这一主体进行操作的。此模块的数据库设计以及表的创建见于3.3.1和3.3.2节。而在article模块，只进行了博客详情的展示，博客评论的展示两部分，实现过程见4.2.1，4.2.2节。

4.2.1 博客展示

博客是本系统主要的模块之一，因此博客在许多页面都有展示。进入首页，首页的左侧中部面板都是用来展示博客的，这部分展示了博客的发表时间，作者，阅读次数，评论数，所属分类，标签，这些文章相关信息，以及长度为30字符的部分文章正文内容，对其采用了分页显示，每页7篇，点击标题行可查看详细信息，如果是当前用户发表则提示“这篇文章是我发的”，如图4.14所示。



图 4.14 首页博客展示

这种缩略图运用场景还有好几处，分别是归档时缩略信息，按照标签，分类，或者个人中心“我的博客”，都是按照这种缩略信息进行展示，点击标题后详情页面如图4.15，4.16所示。

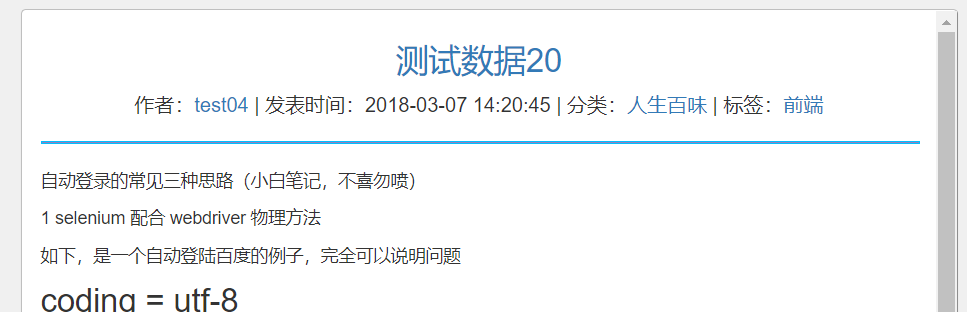


图4.15 博客详情头部

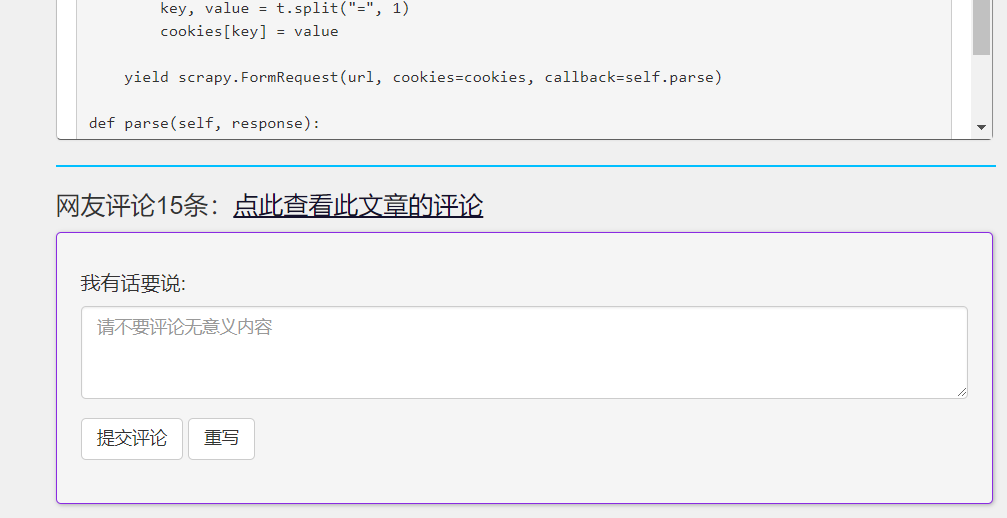


图 4.16 博客详情底部

4.2.2 评论

如图 4.16 所示，博客详情页面显示评论条数，并提供了一个进行评论的文本框，但是需要验证的登录状态才能提交评论成功，以及一个查看此文章已有评论的按钮，点击可查看已有评论，页面如图4.17所示。

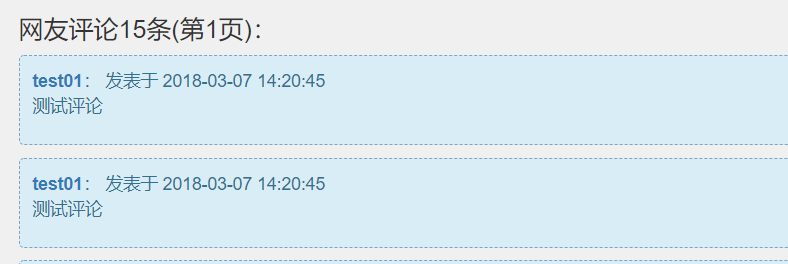


图 4.17 博客评论

评论的页面模板，后台逻辑类似于4.1.7节博客发表过程，其中也需要中间件验证，表单数据合法性验证，数据库更新等流程，此处不再重复说明。

4.3 文件模块的实现

文件中心对于文件的展示已经在4.1.8节做过说明，暂时为每个账号提供了500M的存储空间，暂时不支持已经上传文件的删除操作。此处则对文件的上传，下载原理作为重点进行论述，见于4.3.1和4.3.2节。

4.3.1 文件上传

文件的上传入口只有一个，那就是文件中心的“选择文件”按钮。上传时先点击“选择文件”按钮，会打开本地文件，选择一个本地文件进行打开，然后点击页面“上传”按钮可发起上传请求，服务器会对选择的文件合法性进行判断。可上传的文件目前只有一些常见的文档和图片，如图4.18所示。



图 4.18 文件上传流程

接下来我将详细介绍文件上传的原理。

（1）第一，模板页面设计。在文件中心的“选择文件”按钮实际上是一个表单，表单的关键代码为：

<form method="post" action="/file/upload" enctype="multipart/form-data">

<input type="file" name="upfile" style="display: inline;">

……

</form>

上边展示了文件上传在模板页面的代码，这个按钮实际上是一个表单，和普通的文本输入框差不多，他的请求方式为post，请求路径为“/file/upload”，只不过相比于普通的表单来说添加了一个enctype属性，其值为”multipart/form-data”，然后有一个input标签，他的“type”属性值设为“file”，规定这种格式的表单，就可以生成一个能够读取本地文件进行上传的表单。这个表单在发送请求时将文件数据以流的方式传递给后台，在后台配置”/file/upload”路由的视图函数来处理文件数据即可。

（2）第二，后台逻辑。在前端我们将文件通过一个特殊表单传递了过来，在后台我们取出文件的方法是这样的：

file = request.FILES.get('upfile')

这样的用法非常类似与在后台从取出请求中的参数，使用非常简单，只不过取出参数是从request的请求方式名称类中提取，而此处则是在request.FILE中提取。通过这种方式，我们不仅可以提取出文件数据，还可以提取文件的格式，大小这些元数据。提取出这些值以后，我们将要用这些值来和settings.py中配置的一些信息进行对比，这样也是对于文件数据合法性进行校验，比如单个文件的大小限制为20M以内，用户存储空间为500M，以及限制文件类型等。这些配置，以及限制作用在settings.py中进行配置。

后台经过上边代码中的各项校验之后，进行数据库更新，在数据库中记录文件名，上传者id，上传时间，文件存储的路径。其中文件存储路径是由settings.py中的配置决定总的路径，即和其他各子模块同级的”files”文件夹下，路径还有两部分组成，一是当前登录的账号，然后是文件的分类，分类目前只有img和doc两种，然后是文件。比如说”zhangsan”这个账号上传了一个”123.txt”的文本文件，则后台会将其判定为“doc”文件，他在服务器磁盘创建的完整路径就是：项目/files/zhangsan/doc/123.txt，加入中间的文件夹如果不存在，后台会自动创建。

（3）第三，文件存储。文件上传成功之后，数据库只存储了文件的位置这些元信息，而文件本身的存储则由另外一个函数进行，关键代码：

def save\_file(file, type\_, imgs\_dir, docs\_dir):   
 if type\_ == 'imgs':  
 filepath = os.path.join(imgs\_dir, file.name)  
 else:  
 filepath = os.path.join(docs\_dir, file.name)  
 with open(filepath, 'wb') as fb:  
 # 文件分片读取,每次2048字节，以免占用太多内存  
 from vshare.settings import UP\_FILE\_CHUNK\_SIZE  
 for chunk in file.chunks(UP\_FILE\_CHUNK\_SIZE):  
 fb.write(chunk)  
 return filepath

此处写了一个分片写文件的函数，传入文件对象，文件类型，以及准备好的存放文件的目录，此处从配置文件读取文件分片的大小，这个大小是可以自定义的，因为文件如果太大，一行对文件进行读写操作将会极大的浪费内存，更严重的还可能会让服务器崩溃，因此对文件进行分片存储，每次写入2k直到文件写完毕。这样文件就保存在了服务器磁盘中，如图4.19所示。

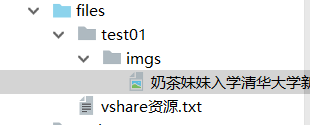


图4.19 文件在服务器的存储

说明，其中有一个名为”vshare资源.txt”的文本文件，直接存放在files下，这个文件是一个对本网站进行推广的广告性质文件，在用户进行下载任意文件时，会和目标文件一起下载给用户，详情见4.3.2节。

4.3.2 文件下载

文件下载的入口有很多处，文件中心的文件列表，点击文件名即可下载对应文件；归档信息中按照分类展示的文件列表，依然是点击文件名下载，还有一处就是首页右侧部分“TOP10文件”这一板块，如果未登录则隐藏，否则展示下载量最多的前10个文件，点击文件名也可直接下载，文件的下载需要验证登录状态。文件下载的流程如图 4.20 所示。

文件下载过程实际上就是对数据库文件的读操作，详细情况参考file/views.py中的代码，部分关键代码如下：

with open(path, 'rb') as fr:

while True:

chunk = fr.read(chunk\_size)

if chunk:

yield chunk

else:

break



图 4.20 文件下载流程

（1）首先根据请求中的参数去数据库查找文件位置.

（2）未查找到则返回提示，查找到则拿着文件位置去服务器磁盘寻找文件，找到文件则读取文件到内存中，此处读取依然采用分片读取，分片读取采用的是Python中文件对象的read()函数，传入一个字节数大小就可以每次读取指定字节数。

（3）同时读取图4.19中的“vshare资源.txt”文件，然后将这两个文件进行打包成zip压缩包，这个压缩包存放在名为“zips\_tmp\_file”的临时文件夹中。

（4）随后将这个.zip文件数据以流的形式返回给浏览器，最后删除这个临时文件。文件下载后存储的位置和浏览器设置有关，下载的形式是附件，如图4.20所示。



图 4.21 下载下来的文件

解压这个文件，可以看到刚才需要的下载的文件和一个“vshare资源.txt”，下载下来的文件可以正常使用，“vshare资源.txt”文件的内容如图4.22 所示。



图 4.22 vshare资源.txt

4.4 首页模块的实现

首页模块设计第一部分，首页欢迎信息。如果用户在访问首页时已经成功登录，则首页导航条显示“欢迎xx”信息，xx指的是已登录的账号，点击这个欢迎信息会进入个人中心，欢迎信息后面还会有一个“注销登录”按钮，点击可以退出当前账号的登录状态。倘若进入首页时用户未登录，则显示“登录/注册”按钮，为用户提供登录的入口。此部分由4.1.2节提到的logined\_user\_processor的全局变量完成，如果全局变量存在则表名用户已登录，从session中取出已登录user对象使用，否则就提供登录入口。

首页模块设计第二部分，归档信息。此部分位于首页右侧，由4.5节archive模块完成，此处不做叙述。此处介绍全局上下文的使用方法，在mainpage模块下新建logined\_user\_processor.py文件，文件内容就是定义一个函数，这个函数返回一个字典，字典的键为模板中变量的名字，值为已登录user对象，未登录则是空对象。然后去settings.py中如图 4.23 所示配置。

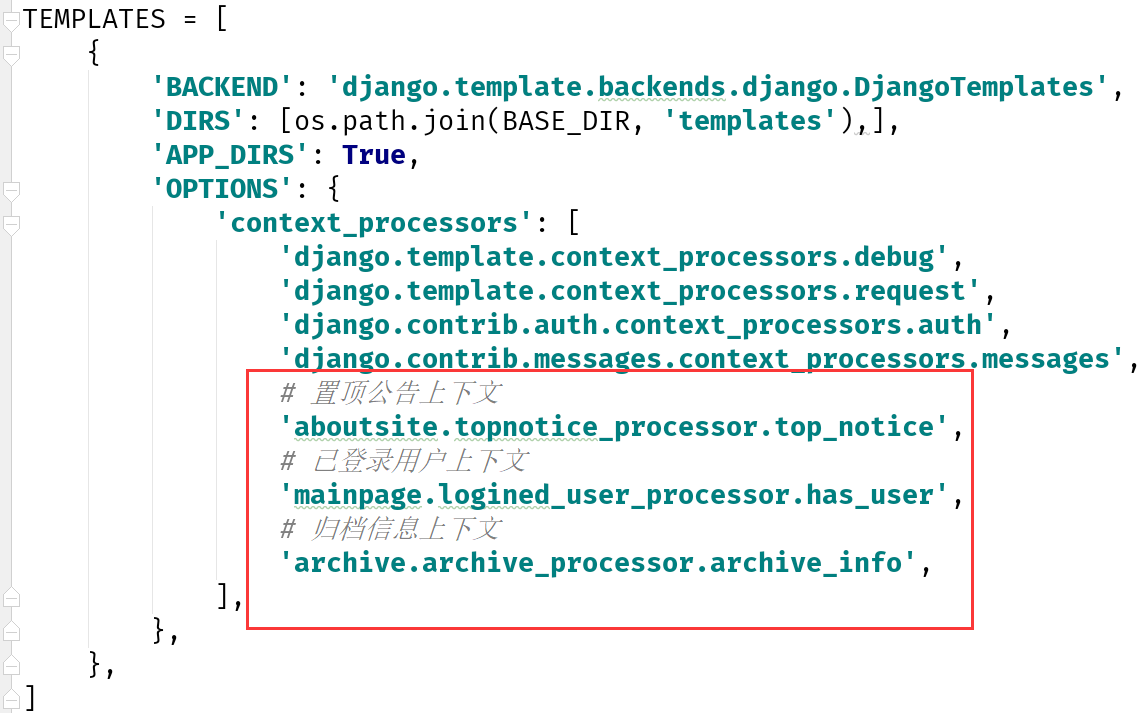


图 4.23 全局上下文配置

如上图所示，红色部分为本项目中定义的所有全局上下文，“OPTIONS”其他部分则是Django模块自带的。全局上下文在此处注册之后方能生效，注册格式“模块名”+“包名（.py文件名）”+”函数名”。注册这个全局上下文之后，在对任意一个请求进行响应时会携带这个函数返回的键值对给模板，也就是说函数返回的变量会被传递给所有的html文件，这就是全局上下文的意义。

首页的第三部分也就是左侧中间部分，已经在4.2.1节展示了最终页面，他的主要逻辑就是从数据库查询出已有的博客，按照分页数进行展示，分页这一逻辑将在4.7.2节详细介绍，此处没有携带分页参数则默认第一页。

4.5 归档模块的实现

归档模块分为三部分，分别是对博客进行分类的归档，标签的归档，以及对文件分类的归档，主要逻辑就是在后台查询数据库，按照关键字段进行查询，然后将查询到的结果返回到前端页面，前端页面根据查询到的结果集进行遍历出来，就得到了归档后的页面，以按照分类归档博客为例，页面效果如图4.24 所示，实现过程参考3.4.3节。



图 4.24 分类归档页面

dic['cat\_'] = Category.objects.get(is\_delet=0, id=cat\_id)

queryset = Article.objects.filter(is\_delet=0, category\_id=cat\_id)

在归档页面提供了类别名，归档依据，搜索到的博客按照分页的方式展示。按照标签归档的方式与此相同，以上是归档的关键代码，实际就是对于数据库的查询操作，图4.25是文件归档效果。



图 4.25 文件归档页面

4.6 网站相关模块的实现

4.6.1 网站访问统计

网站统计的逻辑也很简单，判断浏览器是否为直接访问首页，然后从请求中提取访问网站的ip地址，数据库中存储当前访问ip，时间信息。因此，这一功能在mainpage的模块中进行完成，在处理首页访问请求时进行数据库visit\_document表的更新。这一统计信息目前暂未发挥其余作用，仅用来把控网站流量信息。统计信息如图4.26 所示，后台逻辑如图4.27 所示。

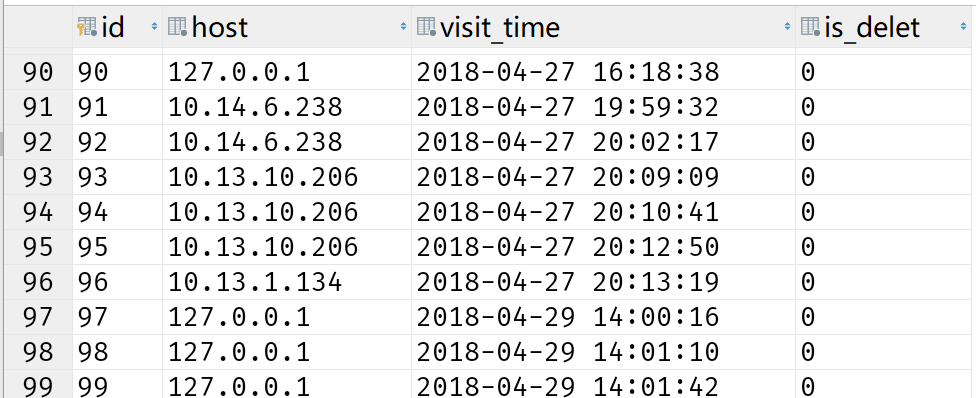


图 4.26 首页访问统计



图 4.27 网站访问记录流程

如图 4.27，其中判断是否为首页第一页，查看访问路径即可。在访问这个网站时，假如是通过ip直接访问，则请求路径为“/”，否则为“/mainpage”或者“/mainpage/2”这样的路径，因此只要请求路径为“/”就判定为首次访问整个网站。响应过程参考4.1.1节。

4.6.2 网站留言板

网站留言板的作用是为网站建设集思广益，因此为游客和会员都提供了进行留言的入口。如果检测到当前已经有账号登录，则显示最近的10条留言并提供一个文本框，用户只需输入留言内容即可进行留言，因为用户已经登录，恶意灌水的可能性比较小，需要为会员提供便捷的操作方法，因此用户无需像游客一样填写那么多信息。会员入口如图4.28所示。

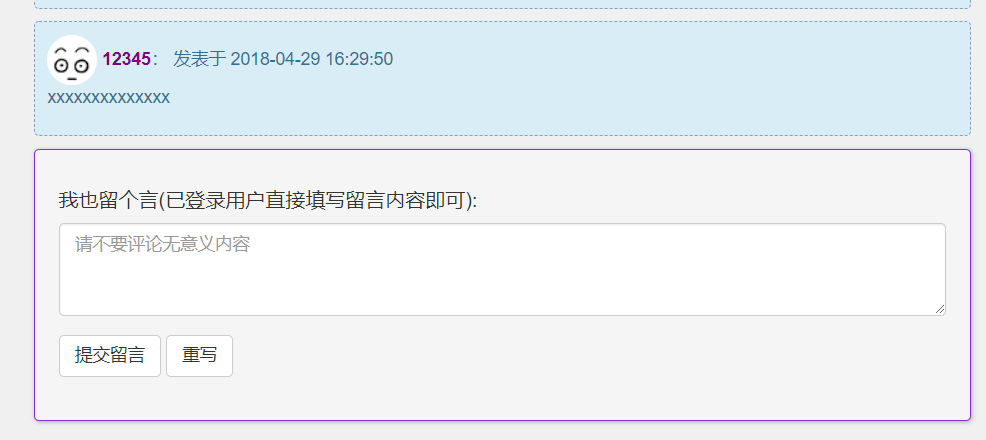


图 4.28 网站留言会员入口

相比于已登录会员，游客也可以留言，但是加入留言也像会员那样不需花费太大精力，则有可能遭到恶意灌水，因此游客留言较为繁琐，游客留言入口如图4.29 所示。



图 4.29 游客留言入口

如图4.29，游客留言时，在前端页面会进行昵称，邮箱，留言内容的判空操作，有一项为空就不能提交，不仅如此，邮箱也必须有邮箱的格式，验证码则会进行后台逻辑的校验。关于验证码的原理将在4.7.3节详细说明。网站留言提交流程如图4.30 所示。



图 4.30 留言提交过程

用户填写留言输入框完成后，点击“提交”按钮将会发起一个post方式的留言请求，服务器从请求中提取用户填写的信息进行各种合法性校验，全部通过则将这些信息写入数据库，否则提示错误原因。响应过程请参考4.1.1节，验证码验证过程参考4.8.3节。

4.6.3 关于

此部分入口位于首页导航条“关于”按钮，点击可访问。这一部分并没有太多后台逻辑，当收到对应请求时将静态页面返回即可。这部分的内容是我的一份简历，主要就是为了推销我自己，网页的内容如图 4.31 所示。



图 4.31 关于

4.7 模板模块的实现

模板模块不涉及到逻辑，它仅仅是一个文件夹，里面保存了各个网页所需要用到的html代码，在响应各个请求时，读取此处的模板返回给浏览器。此处的文件都是静态的超文本文件，并不进行详细说明

4.8 其他技术要点的设计实现

以上章节按照模块分别进行了说明，但是其中用到了一些技术要点并没有详细介绍，这些技术由于有很多地方需要用到，我将它们共同封装在了一个工具模块“utils”当中，使用时只需要导入这个模块，调用其中的函数即可，这种做法也是实际开发中常用的一种设计思路。这些将在接下来的4.8.1，4.8.2，4.8.3节进行说明。

4.8.1 格式化输出时间

对于数据库的设计，许多表都设计了时间字段，因此如图4.32中getnow文件，这个文件专门用于提取操作系统的时间进行格式化输出，代码部分不做介绍，下面由表4.3介绍其中的函数及其作用。

表4.3 格式化输出时间

|  |  |
| --- | --- |
| 函数 | 作用 |
| now() | 输出完整24时制时间，格式”年年年年-月月-日日 时时:分分:秒秒” |
| date\_time() | 输出年月日，格式同now() |
| time\_time() | 输出时分秒，格式同now() |
| this\_year() | 以字符串格式输出年份的四位数 |
| this\_month() | 以字符串格式输出月份两位数 |
| this\_day() | 以字符串格式输出日期两位数 |
| this\_hour() | 以字符串格式输出小时两位数 |
| this\_minute() | 以字符串格式输出分钟两位数 |
| this\_second() | 以字符串格式输出秒钟两位数 |

4.8.2 通用分页处理

随着网站的运营，博客，文件，用户的数量都会慢慢增多，当数量多到一定程度，如果直接在数据中查询出所有的记录，那将会变得十分缓慢，当数据量足够大时，甚至会让服务器崩溃，因此我们需要做分页处理，分别查询出每一页的记录返回，而在需要其他页时再查询别的记录，在前端页面提供分页条给用户，这样就可以大大减少数据库和服务器的压力，并且查询速度也不会变慢。

如图4.32中的”page\_breaker.py”文件，其中定义了一个pagebreaker()函数，他接收三个参数，分别是要查询的表中所有的记录即一个queryset，需要获取的页数，每页需要查询的记录条数。他的返回值也有三个，分别是查询到的结果集queryset，当前页所在的页数范围，分页后的页码总数。Pagebreaker利用了Django框架内置的Paginator类，输入和输出接收的queryset都是通过ORM方式查询数据库得到的类似于列表的一个结果集。

4.8.3 验证码的应用



图 4.32 验证码原理

如图4.29所示，验证码以绿色为背景，黑色五位随机数字作为值，在网页上表现为一张图片。这个验证码是由code.py文件生成，生成五位随机数字之后，将他绘制为一张图片，并且图片上添加有干扰线以防被一些工具轻易识别，函数返回验证码字符串以及生成的图片，后台将字符串存入缓存数据库redis，而将图片传给前端页面，当留言提交时则从数据库取出字符串和用户提交的验证码进行对比，正确则继续留言操作，否则提示验证码错误并且重新生成验证码开始刚才的逻辑。验证码原理如图4.32所示。

4.8.4 web 服务器监控

服务器在运行过程中出现宕机，崩溃等各种问题是不可避免的，需要对其进行监控，以便在出现这些情况时及时通知，及时处理，以免造成长时间大面积的网站无法访问的情况。在本系统设计中，我设计了一个简单的监控程序，用来监控我的网站服务情况。

我设计的原理是，根据设定的时间间隔去对首页发起访问请求，如果出现目标计算机积极拒绝，返回的状态码不是2xx的情况，就认为服务器出现异常，此时调用我设定好的短信接口，发送短信到我的手机通知我。

第五章 系统测试

5.1 服务器环境配置

5.1.1 Python环境

第一步，用xshell工具远程连接服务器。

第二步，确保服务器网络畅通。可以使用命令“ping [www.taobao.com](http://www.taobao.com)”来进行测试。

第三步，下载Python3。CentOs系统默认的是python2.7环境，但是我们需要用到python3，因此需要自己安装配置。我使用的是发行版的”Anaconda-4.2.0”，他是一个发行版的python环境，集成了大部分日常需要用到的包。命令如下。

yum -y install wget

wget <https://mirrors.tuna.tsinghua.edu.cn/anaconda/archive/Anaconda3-4.2.0-Linux-x86_64.sh>

中间需要几次确认，输入y确认即可。

第三步，安装Anaconda，使用命令“bash Anaconda3-4.2.0-Linux-x86\_64.sh”即可完成安装。

第四步，配置环境变量。进入“/etc/profile”文件，末尾添加“export PATH=$HOME/bin:/usr/python/bin:/usr/local/bin:$PATH”。保存退出并重启操作系统。

第五步，测试python环境。执行命令“python”出现图5.1所示交互环境则证明python环境搭建成功。

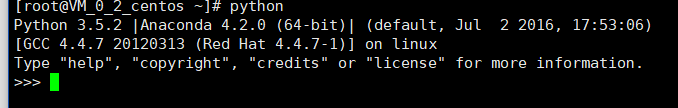


图 5.1 python交互窗口

5.1.2 redis数据库

网络畅通的情况下直接使用以下命令。

wget http://download.redis.io/releases/redis-2.8.17.tar.gz

tar xzf redis-2.8.17.tar.gz

cd redis-2.8.17

make

由于我们的redis和项目文件安装在同一个服务器上，无需更改redis.conf的配置，使用默认的就可以。

下一步，启动redis服务。执行以下命令，出现图5.2 所示提示则说明redis服务正常启动。

cd src/

redis-server



图 5.2 redis服务启动

5.1.3 依赖的Python包

在开发过程中使用到了很多模块，如果不是python标准库中的模块，就要自己安装，在python环境安装好的情况下依次执行以下命令，需要确认的地方确认即可。

pip install Django

pip install Django-redis

pip install markdown2

5.2 项目测试上线

5.2.1准备项目文件

使用xftp工具连接服务器，将调试好的项目整个文件拖拽上传到服务器root目录下，假如登录的是普通用户，则可以将项目文件放置在自己的用户目录。

5.2.2 修改项目配置

vi /root/vshare/vshare/settings.py

将其中的DEBUGE=True 改为DEBUGE=False，并添加配置项，之后保存退出。配置项的作用是：指定本机6379端口运行的redis为缓存数据库，关闭浏览器之后session失效。

5.2.3 后台启动web服务

做完以上配置之后，进入项目中manage.py文件所在目录，执行以下命令：

nohup python manage.py runserver 0.0.0.0:80 –insecure

以上命令是将我们的django项目以正常模式后台运行运行在80端口，这样当我们退出服务器的远程连接之后依然能正常访问网页，并且由于是80端口，默认为web服务，所以直接在浏览器输入地址而不需输入端口号。

5.2.4 访问网页

打开浏览器，地址栏输入服务器的公网ip，比如我的服务器为193.112.207.240，出现图5.3 所示页面则访问成功。



图 5.3 访问网站‘

图 5.3 为博客首页，可以点击页面上的各个按钮，连接进行功能体验，除了右上角“站内搜索”尚未开放功能，其余页面均可正常使用。

第六章 总结

6.1 项目总结

本次设计，完成了前端页面，后台逻辑，以及对数据库的设计，虽然也存在很多的问题，但是基本上是完成了一个完整的web项目。接下来我就围绕这个web项目的三个层面进行总结。

数据库层，使用了关系型数据库Sqlite3，他类似于MySQL数据库，是实际开发中一种常用的关系型数据库。关系型数据库最为重要的一个特点是表与表之间存在着主外键关系，分析需求设计数据库是至关重要的一步。比如在此项目中，我的第一步就是分析需求，然后设计表，后边开发的整个过程是没有对数据库进行任何更改的，而当我后期发现数据库某些字段设计不太合理时，更改数据库的代价太大了，因此并没有更改表的结构。在实际开发中也是如此，一旦确定下数据库设计，是不允许对表结构进行变更的，由此可见数据库设计的重要性。关系型数据库依然是如今开发的主流数据库，通常关系型数据库用来做项目稳定长期的数据保存，因此掌握ORM方式操作数据库也是很有必要的，世纪的开发中就是使用这种方式。在此项目中，我还用到了非关系型数据库redis用来做缓存数据库，实际开发也是这样的，使用键值对存取的数据库进行缓存数据的存储，速度快，支持高并发，使用方便灵活。而非关系型数据库通常情况下并不适合做永久的，长期的数据存储，因为存在过期时间设定。由此也可以看出，至少掌握一种关系型数据库，一种非关系型数据库也是很有必要的，他们各自都有自己的用途。

视图层，即提供给用户浏览的架构设计。本次设计实际上还是十分粗糙的，使用了Bootstrap，jQuery两个框架进行页面的设计与实现，以及自己写的一些CSS样式和JavaScript脚本对网页与用户的交互情况进行控制。页面比较简陋，用户体验不是太好，但是实现了基本功能。当前开发岗位有专门做试图层的前端开发，与之相比我确实还有很大差距。不过对于前端的东西，即使做后台开发也必须有一定的掌握程度，因为前端并不是网站独立的部分，经过后台逻辑的连接才会赋予它灵魂，让他有生命，而非一张静态页面。后台逻辑向前端传递参数，进行渲染是连接后台于前端的关键步骤，而这一步骤通常由后端工程师完成，因此后端工程师更应该对前端有所了解。

业务逻辑层是Web项目最大的工程，也是本次设计的侧重点。业务逻辑层负责连接数据库层和视图层，并且整个项目的业务逻辑都由他来实现。业务逻辑层做的事情概括来说就是三件事：第一，接收前端页面用户在网页上的操作。第二，根据用户操作设计业务然后返回响应。第三，需要存储，更改，查询数据的地方由后台来连接数据库进行操作，给前端页面结果反馈。本次设计就以后台设计为重点，数据库设计和视图层设计为次要任务，完成了一个完整的Web项目，网站的用户体验，以及网站的性能暂时并不深入。

6.2 项目拓展

尽管本次设计完成了基本功能，但是任然有很多地方需要完善，对其功能进行拓展，或者对其性能进行提升，注入以下这些方面，都是可以进行拓展的地方。

其一，网站首页站内搜索功能尚未完成。随着博客以及文件数量的增多，有时候要快速找到目标就需要用到站内搜索。做站内搜索不能直接对数据库进行查询，而是需要使用搜索引擎，当数据库记录非常多之时，采用数据库查询将会非常缓慢，难以实现快速查找，而搜索引擎则会对指定字段建立索引，查找起来会更加快捷。我已经在前端页面添加了功能框，但是还未开放，这一功能将在之后逐渐开放。

其二，网站账号注册过于简单，数据库存储账号信息时未加密。我在提供账号注册接口时，仅仅进行注册资料填写，如果加入短信验证，邮件验证这样的机制，注册起来将会更加安全，网站资源也会更加安全。我在存储用户注册成功的账号信息时，直接将未加密处理的账号密码等信息存储到了数据库，这样一来开发人员可以轻易知道这些私密信息，并且一旦数据库遭到泄露，将会有很大的安全隐患。在后续的完善中我将添加像密码这样的关键信息加密之后在进行存储这样的机制。

其三，网站留言板信息未提供及时回复功能。网站留言板暂时仅仅对用户的留言进行了存储，并未提供回复用户的功能，这样将会降低用户粘性。后续打算开发一个邮件接口，对用户的建议进行及时反馈。

其四，用户发表博客的编辑器过于简陋，并且存在JS注入的安全问题。目前仅提供了markdown语法的富文本支持，但是并不是每一个人都会使用，因此需要在页面中集成编辑器，为用户提供更加丰富的文本编辑功能。并且，由于需要使用markdown富文本效果，编辑区没有进行过滤的，这样一来用户有可能进行js注入，存在很大的安全隐患，我已经证实，目前确实可以进行JS注入。在后续处理中，我将完善这个编辑器并且对与<script>这样的一些关键字进行处理之后在进行显示，以此减少JS注入的几率。

参考文献

[1] [千锋Unity游戏开发](http://mp.sohu.com/profile?xpt=cHBhZzIwOTEwMGY2NzJmMEBzb2h1LmNvbQ==&_f=index_pagemp_1).Python是什么？可能是最受欢迎的编程. [OL]. <http://www.souhu.com/> a/167256479\_397765

[2] php 中文网. 哪些大公司使用Python. [OL] http://www.php.cn/python-tutorials-88939.html

[3]腾讯云社区. 云计算Python自动化,一些比较有名的网站或应用-云+社区-腾讯云[OL]

<https://cloud.tencent.com/developer/news/211365>

[4] CSDN . TensorFlow和Caffe、MXNet、Keras等其他深度学习框架的对比[OL] https: //blog.csdn.net/xiangz\_csdn/article/details/57074000

[5] PaddlePaddle深度学习API. [OL] <http://staging.paddlepaddle.org/> docs/develop /documentation /fluid/zh/design/motivation/api.html

[6] 赵越；陈志伟；蔡淑. 大数据量科学计算数据的动态Web可视化[J/OL] 现代计算机：上半月版 2012年

[7] 张翀. 超文本下的知识进化 [J/OL] 东华大学学报：社会科学版 2004年

[8] 郭涛；黄铭钧. 社区网络爬虫的设计与实现 [J/OL] 智能计算机与应用 2012年

[9] [Truong的专栏](https://blog.csdn.net/truong). HTTP详解 [OL] <https://blog.csdn.net/Truong/article/details/49778067>

[10] Elizabeth Castro．HTML XHTML CSS基础教程．[M]北京人民邮电出版社，2007.8

[11] 何守才;周龙骧,唐世渭,马应章等《数据库百科全书》[M];上海交通大学出版社 2009-09

[12] 甘胜玉 . 网络内容分析与还原系统的设计与实现 [D/OL] 硕博学位论文 2009年

[13] 万方数据. HTTP协议及其发展 [J/OL] http:// www. wanfangdata. com.cn/details/ detail.do?\_ type=perio&id =dnxx199901001

[14] Django官网. The Django template language [OL] <https://docs.djangoproject.com/en/> 2.0/ref/ templates/language/#the-django-template-language

[15] 数据百科全书. 超文本传输协议 [OL] http:// gongjushu.cnki.net /refbook/detail.aspx? recid = R2011090780000058

[16] 姜晗; 任翠池; 王磊 .基于Cookie和Session的身份认证机制的研究与实现 [J]中国教育技术装备出版社 2014-02-25

[17] 梁捷. Web App的未来 [M/J] 程序员 2012年

[18] 开源中国社区 .超高性能key-value 数据库Redis [OL] <https://www.oschina.net/p/redis>

致谢

本次设计能够顺利完成，感谢指导老师以及同学的大力支持和帮助。现将诸位老师和同学名字列出致以谢意（排名不分先后）。

校内：张涛 湛柏明 周奎 柳祥乐 樊红莉 郝夏燕

校外：张露

同学：杨玉祥

感谢以上老师和同学对我的支持和帮助。

附录：项目源码

由于此项目包含多个模块，以及多种文件，无法在文档中贴出，项目源码请参考附件。