

本科毕业论文

论文题目： ICPC.scnu.edu.cn二期建设

指导老师： 曹霑懋

学生姓名： 王荣瑞

学 号： 20171131046

学 院： 计算机学院

专 业： 计算机科学与技术

班 级： 5班

**目录**

摘要

ABSTRACT

1. 绪论

1.1建设网站的目的和意义

1.2国内外ICPC现状和发展意义

1.3课题主要工作

1. 网站开发相关技术

2.1课题的可行性分析

2.2可行性分析

2.2.1VUE的特点

2.2.2技术可行性分析

2.2.3可靠性和安全性特点

2.3DJANGO框架介绍

2.3.1DJANGO是什么

2.3.2DJANGO技术特点

2.3.3DJANGO后端开发WEB的方式

2.4VUE技术介绍

2.4.1VUE是什么

2.4.2VUE的特点

1. 软件需求分析

3.1功能分析

3.1.1功能划分

3.1.2功能描述

3.2外部接口需求

3.2.1用户界面

3.2.2软件相应接口

1. 软件概要设计

4.1软件总体设计

4.1.1处理流程

4.1.2总体结构与模块外部设计

4.2数据库设计

1. 软件详细设计

5.1软件设计详细说明

5.1.1总体设计

5.1.2登录界面模块描述

5.1.3管理员模块描述

5.1.4主界面描述

结论

参考文献

摘要：随着网络教育的普及，教学网站的建设成了信息化网络建设的迫切需求。而教学网站中更加关注是否具有实际的应用价值和完善的功能，结合上述的研究，按照软件工程的思想，基于django框架设计并实现一个可供用户登录注册、在线测试、在线讨论的ICPC网站。构建一个好的网站，不仅需要提高网站的硬件性能、扩展网站模块功能，更应该注重对计算机系统和网络的防护，网站上的数据库中存储大量重要的信息与资料，需要重点对网站的数据库安全进行研究，采用文件型加密算法对数据库进行加密，提升数据安全性能。

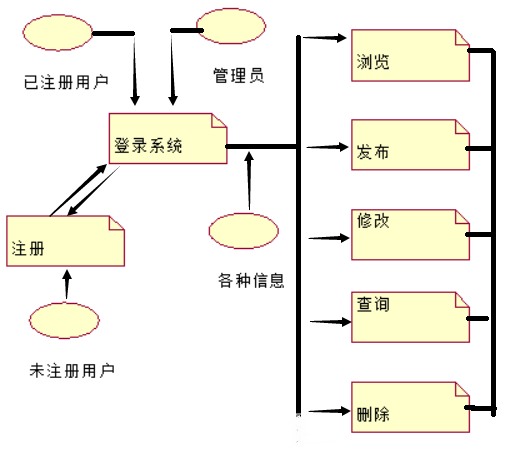
1. 需求分析

针对ICPC.scnu.edu.cn二期建设，需要设计包含了网站的WEB前台和后台所需的所有核心功能：网站首页、在线测试、资料下载、站内新闻和在线讨论。

站内的用户分为三种，分别为未注册用户，已注册用户和管理员。

首先对于未注册用户允许打开网站首页和站内新闻页面，需要查看其他信息必须进行注册并登录。同样的，对于已注册用户登录后也方可以查看在线测试、资料下载、在线讨论页面。管理员输入正确的密码后，进入到管理员界面，该界面包括对网站信息修改、站内新闻管理、资料上传、在线测试管理、用户管理等操作。

主要业务流程

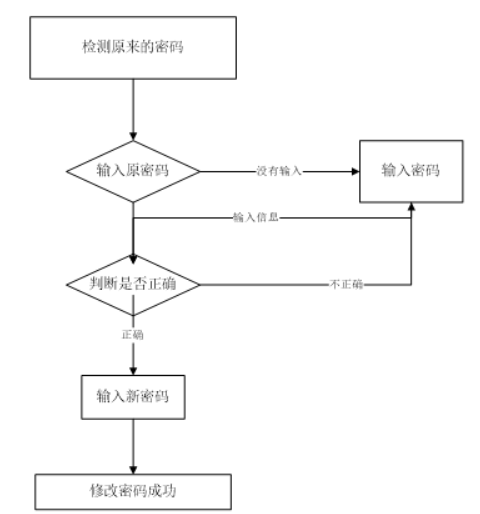


1. 概要设计

系统的处理流程：

模块划分：

修改密码的流程：



接口设计：

后端开发过程：

通过github下载

环境说明

判题机需部署在Linux环境下！

项目使用的环境是：

前端： Ubuntu 18.10 + Nginx

后端： Ubuntu 18.10 + Python 3.7

判题服务器： Ubuntu 18.10 + Python 3.7

判题机： Ubuntu 18.10 + Python 3.7 （必须Linux系统）

爬虫部分： Ubuntu 18.10 + Python 3.7

整个OJ对系统要求不高。1G内存 1核的机器足以。但是部分题目需要大内存，所以推荐使用2G内存 1核的服务器。

准备部署

首先我们把代码clone下来，或者直接在Github上下载下来。

首先我们将代码解压到一个文件夹中（就用LPOJ作为名字吧！）

在linux服务器上面操作

mkdir LPOJ

cd LPOJ

git clone <https://github.com/Linzecong/LPOJ.git>

git可能会出现时间缓慢的问题

可采用git来下载

git clone git://github.com/Linzecong/LPOJ.git

这样就将所需的文件下载下来

文件的目录如下



后端部署开发：

后端的开发比前端的开发要简单很多，因为开发者只需专注于数据的呈现即可，不必关心显示的逻辑。在众多后端框架中，我选择了开发和学习成本较低的Python语言中的Django框架，同时Python语言与我们的判题程序又相辅相成，因此是一个很好的选择。Django是一个开源的Web框架，整体采用MVC的设计模式。但是在本系统中，我们并不需要构建自己的前端页面，我们只关心有哪些数据要交付给前端，因此本系统采用Django中的REST框架来快速构建自己的数据API。其中REST是RESTful的简称。RESTful是一种软件架构风格，它采用http协议，非常简单，任意客户端都能运行。因此我们只需要关心有哪些数据要交付给前端即可。其中REST框架主要分为三个部分，分别是Model,Serializer和View。Model即数据层，定义了数据在数据库中的形式。Serializer即序列化器，其中定义了各种数据库的操作，相当于一个中间层，最后View层决定了哪些数据可以呈现给用户，怎么呈现给用户等等。所以当开发者编写API时，只要着重于实现这三层即可。因为有了RESTful框架，这一切都变得非常简单便利。

Django REST framework 介绍

现在越来越多的网站采用前后端分离技术。在前后端分离的应用模式中，后端仅返回前端所需要的数据，不再渲染HTML页面，不再控制前端的效果。前端用户想要看到什么效果，从后端请求的数据如何加载到前端中，都由前端浏览器自己决定。在前后端分离的应用模式中,前端与后端的耦合度相对较低，我们通常将后端开发的每一视图都成为一个接口,或者API,前端通过访问接口来对数据进行增删改查。因此我们采用它进行我们的后端开发，同时他配套有各种各样的服务，如权限管理，路由管理，限流等等，这些在我们的判题系统中尤为重要。

因此本系统采用Django REST framework 开发，接下来跟着我的步伐，一步一步了解Django是如何在LPOJ中运作的！

安装Python3.7

整个开发在Ubuntu 18.10环境下进行，接下来所有教程均在此环境下进行，如果是Windows，请自行查阅安装方法。

Ubuntu自带Python3.6，经测试3.6一样适合本环境。但是Ubuntu中，Python命令默认是Python2，所以我们可以通过如下命令改变默认值，当然你也可以不改变，然后每次执行Python命令时使用的命令应该是Python3

使用下面的代码可以将默认的编译器设置为python3

sudo update-alternatives --install /usr/bin/python python /usr/bin/python2 100

sudo update-alternatives --install /usr/bin/python python /usr/bin/python3 150

如果需要恢复默认设置使用

sudo update-alternatives --config python

首先要配置环境。

先安装pip，管理python的包

sudo apt install python3-pip

本后端主要用了如下几个库

Django==2.2.1

djangorestframework==3.9.3

django-filter==2.1.0

django-cors-headers==2.5.3

mysqlclient==1.4.2.post1

其中第一个是整个Django框架所需的库，第二个是我们的REST Framework，第三个是用于实现过滤功能的一个框架，第四个是用于实现跨域访问的框架，第五个是访问Mysql所需要的库。

进入到Backend目录并执行如下命令即可。

为了防止出现mysql\_config:not found的错误

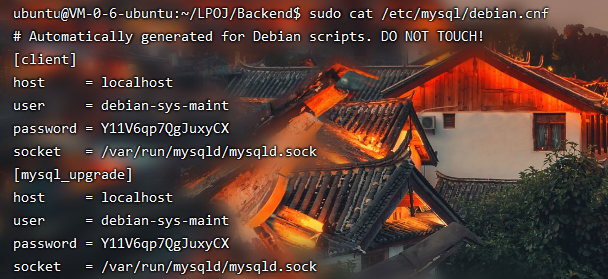
sudo apt-get install libmysqlclient-dev

pip3 install -r requirments.txt

安装mysql

sudo apt-get install mysql-server

找到安装时默认生成的默认账户在终端上输入  sudo cat /etc/mysql/debian.cnf  如下图显示的账号和密码



用上述的账号和密码进行登录

mysql -udebian-sys-maint -p

Enter password:Y11V6qp7QgJuxyCX

这样进入mysql

1)、use mysql;                   #连接到mysql数据库

2)、update mysql.user set authentication\_string=password('123456') where user='root' and Host ='localhost';    #修改密码123456是密码

3)、update user set  plugin="mysql\_native\_password";

4)、flush privileges;

5)、quit;

这样就把账号和密码修改为root和123456

以后可以通过

mysql -uroot -p123456进行登录

当然为了安全性还是建议把密码设置更加复杂一点



顺便补充一点mysql服务的启动和关闭命令 转载 [详解Ubuntu Server下启动/停止/重启MySQL数据库的三种方式(ubuntu 16.04)](https://www.cnblogs.com/nyist-xsk/p/7930035.html" \t "_blank)

启动mysql：

方式一：sudo /etc/init.d/mysql start

方式二：sudo service mysql start

停止mysql：

方式一：sudo /etc/init.d/mysql stop

方式二：sudo service mysql stop

重启mysql：

方式一：sudo/etc/init.d/mysql restart

方式二：sudo service mysql restart

接下来回到数据库，创建一个LPOJ的数据库，并授予root所有权限

CREATE DATABASE LPOJ DEFAULT CHARACTER SET utf8 COLLATE utf8\_general\_ci;

USE mysql;

GRANT ALL PRIVILEGES ON \*.\* TO 'root'@'%' IDENTIFIED BY 'your\_password' WITH GRANT OPTION;

ALTER user 'root'@'%' IDENTIFIED WITH mysql\_native\_password by 'your\_password';

flush privileges;

exit;

sudo nano /etc/mysql/mysql.conf.d/mysqld.cnf

**编辑setting.py**

接下来我们修改一下配置文件，我们进入到Backend/Backend目录下，并修改setting.py文件

cd Backend

nano setting.py

修改数据库的用户、密码和ip地址

对文件进行修改保存时，用到nano编辑器，修改后保存修改并退出的方法：

打开并修改文件：nano settings.py

修改后使用 Ctrl+X，

提示：save modified buffer ...?    ，选择 ：yes

又提示：file name to write ：   ，选择：Ctrl+T

在下一个界面用 “上下左右” 按键 选择要保存的文件名，

 然后直接点击 “Enter” 按键即可保存。

运行后端

在Backend根目录中的manager.py是Django非常重要的一个文件，通过它我们可以实现很多操作，比较常用的操作如下：

makemigrations 将你编写的代码变成sql语句

migrate 将你编写的sql语句同步到数据库中

runserver 运行你的后端

首先将LPOJ的后端代码生成mysql语句，然后我们再同步到数据库中

python manage.py makemigrations

python manage.py migrate

执行完毕后查看数据库，会发现你的LPOJ数据库中多了若干表格，这些都是Django运行所需要的表格。

在运行之前让我们先制作一个超级用户，这在部署文档里面有写

echo "from django.contrib.auth.models import User; User.objects.filter(email=\"admin@example.com\").delete(); User.objects.create\_superuser(\"admin\", \"admin@example.com\", \"admin\")" | python3 manage.py shell

1

最后我们再执行运行语句，以后如果对后端有任何修改，只需先执行

python3 manage.py makemigrations

python3 manage.py migrate

将修改同步到数据库中，然后再通过如下命令运行

python3 manage.py runserver 0.0.0.0:8000

意思是将后端暴露到8000端口中，并监听所有地址的访问。

接下来，我们就可以通过在浏览器中通过localhost:8000访问你的后端了。

如果能访问，证明你的后端已部署成功！

模块说明与开发

在阅读如下教程时，请先自行学习Django REST Framework的一些基本教程

接下在会对一些模块做简单介绍

Backend 保存整个后台的配置文件和路由

blog 博客模块

board 排行榜相关模块

contest 比赛相关模块

judgestatus 提交信息模块

problem 题目模块

user 用户相关模块

wiki Wiki模块

classes 班级管理相关模块

ProblemData 存放题目数据的文件夹

本教程不会对每个模块都做详细介绍，但是会列出各文件的功能，和点出一些自己的实现方法。

Backend

setting.py

此处添加你自己的模块

INSTALLED\_APPS = [

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

'rest\_framework',

'django\_filters',

'judgestatus',

'corsheaders',

'problem',

'user',

'contest',

'board',

'blog',

'wiki'

]

此处填写一下你使用的框架的设置，比如限流设置就是在此处实现。

REST\_FRAMEWORK = {

'DEFAULT\_PAGINATION\_CLASS': 'rest\_framework.pagination.PageNumberPagination',

'DEFAULT\_FILTER\_BACKENDS': ('django\_filters.rest\_framework.DjangoFilterBackend',),

'DEFAULT\_THROTTLE\_CLASSES': (

'rest\_framework.throttling.AnonRateThrottle',

),

'DEFAULT\_THROTTLE\_RATES': {

'anon': '300/m',

'judge': '300/m',

'post': '2000/m',

}

}

此处是一下数据库的设置

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.mysql',

'NAME': 'LPOJ',

'USER': os.environ.get("DB\_USER") if os.environ.get("DB\_USER") else 'root' ,

'PASSWORD':os.environ.get("DB\_PASSWORD") if os.environ.get("DB\_PASSWORD") else 'lpojdatabase',

'HOST': os.environ.get("DB\_HOST") if os.environ.get("DB\_HOST") else 'lpojdatabase',

'PORT': os.environ.get("DB\_PORT") if os.environ.get("DB\_PORT") else 3306,

}

}

此处是跨域访问的设置

CORS\_ORIGIN\_ALLOW\_ALL = True

CORS\_ALLOW\_CREDENTIALS = True

1

2

此处是Session过期设置

SESSION\_COOKIE\_AGE = 60 \* 60 \* 24 # 30分钟

SESSION\_SAVE\_EVERY\_REQUEST = True

SESSION\_EXPIRE\_AT\_BROWSER\_CLOSE = True

1

2

3

urls.py

这个文件主要是用来填写路由信息

注意！！这里列出的是我设计的数据库格式，仅做参考~

注意！！这里列出的是我设计的数据库格式，仅做参考~

注意！！这里列出的是我设计的数据库格式，仅做参考~

Blog

此模块主要用来实现博客相关的API

models.py

model 功能

Banner 首页新闻

OJMessage 留言板信息

Blog 爬虫得到的博客条目

Banner

属性 功能 类型 说明

msg 新闻内容 CharField

time 新闻时间 DateField auto\_now

OJMessage

属性 功能 类型 说明

username 留言用户 CharField

msg 留言内容 CharField

time 留言时间 DateField auto\_now

rating 用户留言时的Rating IntegerField 用于实现前端改变颜色

Blog

属性 功能 类型 说明

username 博客所属用户 CharField

title 博客标题 CharField

url 博客地址 CharField

summary 博客的摘要 CharField

time 博客的爬取时间 CharField 爬虫机器人直接保存时间字符串

permission.py

permission 读权限 写权限

UserRatingOnly 有 仅用户或为管理员，并且不能修改Rating

ManagerOnly 有 仅管理员

serializers.py

serializers 序列化字段

BannerSerializer ALL

OJMessageSerializer ALL

BlogSerializer ALL

views.py

视图 查询集合 可过滤字段 权限 可分页 可搜索

BannerView ID倒序 (‘time’) ManagerOnly 是 否

OJMessageView ID倒序 (‘username’, ‘time’) UserRatingOnly 是 否

BlogView ID倒序 (‘username’, ‘time’) ManagerOnly 是 否

urls.py

视图 访问路由

BannerView http://localhost:8000/banner/

OJMessageView http://localhost:8000/ojmessage/

BlogView http://localhost:8000/blog/

Board

此模块主要用来实现排行榜和设置相关的API

models.py

model 功能

SettingBoard 保存OJ的设置

Board 用户爬虫需要的一些信息

DailyBoard 用户每天的AC题目数统计

TeamBoard 队伍Rating信息（该功能暂时弃用）

DailyContestBoard 队伍排位赛的一些信息，用于备份，作用不大

SettingBoard

属性 功能 类型 说明

schoolname 学校名称 CharField 用于显示在排行榜上的学校名字

ojname OJ名称 CharField OJ左上角的标题

openwiki 是否开启Wiki BooleanField

openlanguage 开启的语言 CharField

openoi 是否开启OI判题模式 BooleanField

openstatus 是否开启代码 查看 BooleanField

openvisitor 是否开启游客访问 BooleanField

openregister 是否开启注册 BooleanField

openselfstatus 自己是否能查看自己的代码 BooleanField

Board

属性 功能 类型 说明

username 用户名 CharField Primary\_key

classes 班级 CharField

number 学号 CharField

OJCount 爬虫的OJ数 IntegerField 无用的字段（历史遗留）

OJ 爬虫的OJ CharField 爬虫机器人识别每个账号对应的OJ用，中间用竖线隔开

account 爬虫的账号 CharField 该用户OJ的账号名称，中间用竖线隔开，与OJ字段一一对应

acnum 每个OJ的AC数 CharField 中间用竖线隔开，与OJ字段一一对应

submitnum 每个OJ的提交数 CharField 中间用竖线隔开，与OJ字段一一对应

blogaddress 爬虫用的博客地址 CharField 用的是RSS地址

DailyBoard

属性 功能 类型 说明

username 用户名 CharField

account 当天的总AC数量 IntegerField 每天仅统计一次，用于生成表格

collecttime 采集时间 DateField

TeamBoard

属性 功能 类型 说明

teammember 队伍名称 CharField

score 该队伍得分 IntegerField 由爬虫模块计算并更新

collecttime 采集时间 DateField

DailyContestBoard

属性 功能 类型 说明

contestdate 比赛日期 DateField

teammember 队伍名称 CharField

problemcount 题目数量 IntegerField

wronglist 错误的题目 CharField 中间用竖线隔开，如 A|B

wrongtime 错误的次数 CharField 与wronglist对应，如 1|2

aclist AC的题目 CharField 中间用竖线隔开，如 A|B

actime AC的时间 CharField 与aclist对应，格式为00:00:00

permission.py

permission 读权限 写权限

ManagerOnly 有 仅管理员

serializers.py

serializers 序列化字段

SettingBoardSerializer ALL

BoardSerializer ALL

DailyBoardSerializer ALL

TeamBoardSerializer ALL

DailyContestBoardSerializer ALL

views.py

视图 查询集合 可过滤字段 权限 可分页 可搜索

SettingBoardView ALL 无 ManagerOnly 否 否

BoardView ALL (‘username’,) ManagerOnly 是 否

DailyBoardView 最近10天，正序 (‘username’, ‘collecttime’) ManagerOnly 是 否

TeamBoardView ALL (‘teammember’, ‘collecttime’) ManagerOnly 是 否

DailyContestBoardView ALL (‘contestdate’, ‘teammember’) ManagerOnly 是 否

urls.py

视图 访问路由

SettingBoardView http://localhost:8000/settingboard/

BoardView http://localhost:8000/board/

DailyBoardView http://localhost:8000/dailyboard/

TeamBoardView http://localhost:8000/teamboard/

DailyContestBoardView http://localhost:8000/dailycontestboard/

Classes

此模块主要用来实现班级相关的API

models.py

model 功能

ClassStudentData 学生与班级关系的基本信息

theClasses 班级的基本信息

ClassStudentData

属性 功能 类型 说明

studentUserName 学生用户名 CharField

studentNumber 学生学号 CharField

className 学生所在班级 CharField

studentRealName 学生真实姓名 CharField

theClasses

属性 功能 类型 说明

className 班级姓名 CharField

classSize 班级总人数 CharField

canjoinclass 班级是否开放加入 CharField

permission.py

permission 读权限 写权限

ManagerOnly 有 仅管理员

serializers.py

serializers 序列化字段

ClassDataSerializer ALL

ClassStudentDataSerializer ALL

views.py

视图 查询集合 可过滤字段 权限 可分页 可搜索

ClassDataView ALL (“className”,) 无 否 是

ClassStudentDataView ALL (‘studentUserName’,‘studentNumber’,‘className’,‘studentRealName’) 无 否 是

ClassDataAPIView 用于添加班级 无 无 否 否

DeleteClassDataAPIView 用于删除班级 无 无 否 否

ClassStudentDataAPIView 用于学生加入班级 无 无 否 否

QuitClassAPIView 用于学生退出班级（暂时弃用） 无 无 否 否

urls.py

视图 访问路由

ClassDataView http://localhost:8000/classes/

ClassStudentDataView http://localhost:8000/classStudent/

ClassDataAPIView http://localhost:8000/ADDclasses/

ClassStudentDataAPIView http://localhost:8000/AddClass/

DeleteClassDataAPIView http://localhost:8000/DeleteClass/

QuitClassAPIView http://localhost:8000/QuitClass/

Contest

此模块主要用来实现比赛相关的API

models.py

model 功能

ContestInfo 比赛的基本信息

ContestAnnouncement 比赛中的通知

ContestProblem 比赛包含的题目

ContestBoard 比赛排行榜信息

ContestComment 比赛提问

ContestTutorial 比赛题解

ContestRegister 比赛注册的用户

ContestRatingChange 比赛的积分变化信息

ContestComingInfo 各大OJ近期的比赛汇总，由爬虫机器人收集

ContestChoiceProblem 比赛的选择题

StudentChoiceAnswer 学生选择题的答题

ContestInfo

属性 功能 类型 说明

creator 比赛举办者 CharField

oj 举办的OJ CharField 一般都是LPOJ

title 比赛标题 CharField

level 比赛难度 IntegerField 1~5

des 比赛说明 CharField

note 比赛提示 CharField

begintime 比赛开始时间 DateTimeField

lasttime 比赛持续时间 IntegerField 以秒为单位

type 比赛类型 CharField ACM/Rated等等

auth 比赛权限 IntegerField 1 公开 2 私有 0 保护(需注册)

clonefrom 是否属于克隆 IntegerField 非克隆的为-1，否则保存的是克隆的ID

ContestAnnouncement

属性 功能 类型 说明

contestid 比赛ID IntegerField

announcement 比赛提示信息 CharField

ContestProblem

属性 功能 类型 说明

contestid 比赛ID IntegerField

problemid 题目ID CharField

problemtitle 题目标题 CharField 用于显示在比赛中的题目标题

rank 题目顺序 IntegerField 用数字表示本题目顺序

ContestBoard

属性 功能 类型 说明

contestid 比赛ID IntegerField

username 该提交的用户名 CharField

user 该提交的用户昵称 CharField

problemrank 提交的题目序号ID IntegerField 是题目顺序的编号

type 该提交最终的结果 IntegerField 1 AC， 0没AC算罚时，-1没AC不算罚时

submittime 提交时间 BigIntegerField 提交时间，以豪秒为单位

submitid 该提交的ID IntegerField 用于rejudge

rating 该提交用户的Rating IntegerField 用于前端改变颜色

ContestComment

属性 功能 类型 说明

contestid 比赛ID IntegerField

user 该提交的用户名 CharField

title 提问标题 CharField

problem 提问的题目 CharField

message 提问的信息 CharField

huifu 回复的信息 CharField default=“No respones”

time 提问的时间 DateTimeField auto\_now

rating 该回复用户的Rating IntegerField 用于前端改变颜色

ContestTutorial

属性 功能 类型 说明

contestid 比赛ID IntegerField

value 题解内容 TextField 支持MarkDown

ContestRegister

属性 功能 类型 说明

contestid 比赛ID IntegerField

user 注册的用户名 CharField

rating 该回复用户的Rating IntegerField 用于前端改变颜色

ContestRatingChange

属性 功能 类型 说明

contestid 比赛ID IntegerField

contestname 比赛标题 CharField

contesttime 比赛的时间 CharField

user 本次变化的用户名 CharField

lastrating 上次Rating IntegerField

ratingchange Rating变化量 IntegerField

currentrating 当前Rating IntegerField

ContestRatingChange

属性 功能 类型 说明

ojName 比赛所在的OJ CharField

link 比赛链接 CharField

startTime 比赛开始时间 BigIntegerField 毫秒为单位

endTime 比赛结束时间 BigIntegerField 毫秒为单位

contestName 比赛名称 CharField

ContestChoiceProblem

属性 功能 类型 说明

ContestId 比赛ID IntegerField

ChoiceProblemId 比赛选择题ID CharField

rank 选择题顺序 IntegerField

StudentChoiceAnswer

属性 功能 类型 说明

username 用户名 CharField

realname 真实姓名 CharField

number 学号 CharField

contestid 比赛ID CharField

answer 学生作答的答案 CharField

score 得分 IntegerField

permission.py

permission 读权限 写权限

ManagerOnly 有 仅管理员或clone的比赛已结束

UserRatingOnly 有 仅用户（username）或为管理员，或为PUT

UserRatingOnly2 有 仅用户（user）或为管理员，或为PUT

serializers.py

serializers 序列化字段

ContestAnnouncementSerializer ALL

ContestTutorialSerializer ALL

ContestBoardSerializer ALL

ContestCommentSerializer ALL

ContestInfoSerializer ALL

ContestProblemSerializer ALL

ContestRegisterSerializer ALL

ContestRatingChangeSerializer ALL

ContestComingInfoSerializer ALL

views.py

视图 查询集合 可过滤字段 权限 可分页 可搜索

ContestAnnouncementView ALL (“contestid”,) ManagerOnly 是 否

ContestTutorialView ALL (‘contestid’,) ManagerOnly 是 否

ContestBoardView ALL (“contestid”,“username”,“problemrank”,“type”,) UserRatingOnly 否 否

ContestCommentView ALL (“contestid”,“problem”,) UserRatingOnly2 是 否

ContestInfoView ID倒序 (“begintime”, “level”, “type”,“title”,) ManagerOnly 是 (‘title’,)

ContestComingInfoView 开始时间正序 无 ManagerOnly 是 否

ContestProblemView rank字段正序 (‘contestid’,) ManagerOnly 是 否

ContestRegisterView ALL (‘user’, “contestid”) UserRatingOnly2 是 否

ContestRatingChangeView 比赛时间倒序 (‘user’, “contestid”) ManagerOnly 是 否

CurrentTimeView 当前时间 无 无 否 否

StudentChoiceAnswerView ALL (‘username’,‘contestid’) ManagerOnly 否 否

ContestChoiceProblemView ALL (‘ContestId’,‘ChoiceProblemId’, “rank”) ManagerOnly 否 否

urls.py

视图 访问路由

ContestAnnouncementView http://localhost:8000/contestannouncement/

ContestTutorialView http://localhost:8000/contesttutorial/

ContestBoardView http://localhost:8000/settingboard/

ContestCommentView http://localhost:8000/contestcomment/

ContestInfoView http://localhost:8000/contestinfo/

ContestComingInfoView http://localhost:8000/contestcominginfo/

ContestProblemView http://localhost:8000/contestproblem/

ContestRegisterView http://localhost:8000/contestregister/

ContestRatingChangeView http://localhost:8000/contestratingchange/

CurrentTimeView http://localhost:8000/currenttime/

StudentChoiceAnswerView http://localhost:8000/conteststudentchoiceanswer/

ContestChoiceProblemView http://localhost:8000/contestchoiceproblem/

Judgestatus

此模块主要用来实现提交信息相关的API

models.py

model 功能

JudgeStatus 提交判题的信息

CaseStatus 每一个case的情况

JudgeStatus

属性 功能 类型 说明

user 提交的用户 CharField

oj 提交的OJ CharField 用于VJudge

problem 提交的题目 CharField

result 提交的结果 IntegerField 具体内容查阅Judger文档

time 时间占用 IntegerField

memory 内存占用 IntegerField

length 代码长度 IntegerField

language 提交语言 CharField

submittime 提交时间 DateTimeField auto\_now

judger 判题的判题机 CharField

contest 该提交所属的比赛 IntegerField

contestproblem 该提交所属的比赛的题目 IntegerField

code 提交的代码 TextField

testcase 在哪个样例出错 CharField

message 额外信息 TextField 保存编译错误信息，运行时错误信息等，同时也作为其他OJ的题目ID，用于VJudge

problemtitle 提交的标题 CharField

rating 用户留言时的Rating IntegerField 用于实现前端改变颜色

ip 提交用户的ip CharField 提交者ip地址

CaseStatus

属性 功能 类型 说明

statusid 所属的提交ID IntegerField

username 所属的用户 CharField

problem 所属的题目 CharField

result 该样例的判题结果 CharField

time 该样例所花时间 IntegerField

memory 该样例所花内存 IntegerField

testcase 样例名称 CharField

casedata 样例输入截取 CharField 非比赛才能查看，Judger中控制

outputdata 样例输出截取 CharField 非比赛才能查看，Judger中控制

useroutput 用户输出截取 CharField 非比赛才能查看，Judger中控制

permission.py

permission 读权限 写权限

ManagerOnly 有 仅管理员

UserRatingOnly 无 仅用户或为管理员，并且不能修改Rating

NoContestOnly 仅用户或比赛结束 有

serializers.py

serializers 序列化字段

JudgeStatusSerializer 不包括代码

JudgeStatusCodeSerializer ALL

CaseStatusSerializer ALL

views.py

视图 查询集合 可过滤字段 权限 可分页 可搜索

JudgeStatusView ID倒序 (‘user’, ‘result’, “contest”, “problem”, “language”) ManagerOnly 是 否

JudgeStatusPutView 仅用于提交代码 无 UserRatingOnly 否 否

JudgeStatusCodeView ALL (‘user’, ‘result’, “contest”, “problem”) NoContestOnly 是 否

CaseStatusView ALL (‘username’, ‘problem’, “statusid”) ManagerOnly 是 否

RejudgeAPIView 用于实现Rejudge 无 ManagerOnly 否 否

urls.py

视图 访问路由

JudgeStatusView http://localhost:8000/judgestatus/

JudgeStatusPutView http://localhost:8000/judgestatusput/

JudgeStatusCodeView http://localhost:8000/judgestatuscode/

CaseStatusView http://localhost:8000/casestatus/

RejudgeAPIView http://localhost:8000/rejudge/

Problem

此模块主要用来实现题目相关的API

models.py

model 功能

Problem 题目的详细信息

ProblemData 题目的简要信息

ProblemTag 题目的标签

Problem

属性 功能 类型 说明

problem 题目编号 CharField primary\_key

author 题目作者 CharField

addtime 题目添加时间 DateTimeField auto\_now

oj 题目的OJ CharField Vjudge用，一般是LPOJ

title 题目的标题 CharField

des 题目的介绍 TextField 支持网页格式

input 输入介绍 TextField 支持网页格式

output 输出介绍 TextField 支持网页格式

sinput 样例输入 TextField 多个样例间用|#)隔开

soutput 样例输出 TextField 多个样例间用|#)隔开

source 题目来源 TextField 如果是Vjudge，填的是OJ对应的题号

time 题目限时 IntegerField

memory 题目内存限制 IntegerField

hint 提示 TextField 支持网页格式

auth 题目权限 IntegerField 1公开 2私密 3 比赛中的题

ProblemData

属性 功能 类型 说明

problem 题目编号 CharField primary\_key

title 题目标题 CharField

level 题目难度 IntegerField 1~5

submission 题目提交数 IntegerField

ac 该类型的数量 IntegerField

mle 该类型的数量 IntegerField

tle 该类型的数量 IntegerField

rte 该类型的数量 IntegerField

pe 该类型的数量 IntegerField

ce 该类型的数量 IntegerField

wa 该类型的数量 IntegerField

se 该类型的数量 IntegerField

tag 题目标签 TextField 中间用竖线隔开

score 题目分数 IntegerField

auth 题目权限 IntegerField 1公开 2私密 3 比赛中的题

oj 题目所在OJ CharField

ProblemTag

属性 功能 类型 说明

tagname 标签名字 CharField unique

count 该标签的数量 IntegerField 暂时弃用

ChoiceProblem

属性 功能 类型 说明

ChoiceProblemId 选择题ID CharField default=-1

des 题目描述 TextField 描述题目

choiceA 选项A TextField

choiceB 选项B TextField

choiceC 选项C TextField

choiceD 选项D TextField

permission.py

permission 读权限 写权限

ManagerOnly 有 仅管理员

AuthOnly 仅比赛中或公开 仅管理员

serializers.py

serializers 序列化字段

ProblemSerializer ALL

ProblemDataSerializer ALL

ProblemTagSerializer ALL

views.py

视图 查询集合 可过滤字段 权限 可分页 可搜索

ProblemView 仅单个查询 (‘auth’,) AuthOnly 否 否

ProblemDataView ID倒序 (‘auth’,‘oj’,) ManagerOnly 是 (‘tag’, ‘title’)

ProblemTagView ALL 无 ManagerOnly 否 否

UploadFileAPIView 用于上传测试数据 无 仅管理员 否 否

filedown 用于下载测试数据 无 仅管理员 否 否

showpic 用于显示图片 无 否 否

urls.py

视图 访问路由

ProblemView http://localhost:8000/problem/

ProblemDataView http://localhost:8000/problemdata/

ProblemTagView http://localhost:8000/problemtag/

UploadFileAPIView http://localhost:8000/uploadfile/

ChoiceProblemView http://localhost:8000/choiceproblem/

filedown http://localhost:8000/downloadfile/

showpic/ http://localhost:8000/showpic/

User

此模块主要用来实现用户相关的API

models.py

model 功能

User 用户的详细信息

UserData 用户的简要信息

UserLoginData 用户的登录信息

User

属性 功能 类型 说明

username 用户名 CharField primary\_key

password 密码 CharField MD5加密后的

name 昵称 CharField

regtime 注册时间 DateTimeField auto\_now

logintime 上次登录时间 DateTimeField auto\_now（暂时弃用，见userlogindata表）

school 学校 CharField

course 专业 CharField

classes 班级 CharField

number 学号 CharField

realname 真实姓名 CharField

qq QQ CharField

email 邮箱 CharField

type 权限 IntegerField 1 普通 2 管理员 3 超级管理员

UserData

属性 功能 类型 说明

username 用户名 CharField primary\_key

ac 用户AC数 IntegerField

submit 用户提交数 IntegerField

score 用户总得分 IntegerField

des 用户介绍 CharField

rating 用户的Rating IntegerField

acpro 用户AC的题目 TextField 中间用竖线隔开

UserLoginData

属性 功能 类型 说明

username 用户名 CharField

ip 用户登录的IP CharField

logintime 登录的时间 DateTimeField

msg 其他额外信息，如浏览器版本等 TextField

permission.py

permission 读权限 写权限

ManagerOnly 有 仅POST或为管理员

UserSafePostOnly 有 仅不修改敏感信息或管理员

UserPUTOnly 无 仅用户

AuthPUTOnly 无 仅管理员

serializers.py

serializers 序列化字段

UserSerializer ALL

UserNoPassSerializer 不包括密码

UserNoTypeSerializer 不包括权限

UserDataSerializer ALL

UserLoginDataSerializer ALL

views.py

视图 查询集合 可过滤字段 权限 可分页 可搜索

UserDataView rating倒序 (‘username’,) UserSafePostOnly 否 否

UserView ID倒序 (‘username’,) UserSafePostOnly 是 否

UserChangeView ALL 无 UserPUTOnly 否 否

UserChangeAllView ALL 无 AuthPUTOnly 否 否

UserLoginAPIView 用于登陆 无 AllowAny 否 否

UserUpdateRatingAPIView 用于更新本地Rating 无 AllowAny 否 否

UserLogoutAPIView 用于登出 无 AllowAny 否 否

UserRegisterAPIView 用于注册 无 AllowAny 否 否

UserLoginDataView ID倒序 (‘username’,‘ip’,) ManagerOnly 是 是

urls.py

视图 访问路由

UserDataView http://localhost:8000/problem/

UserView http://localhost:8000/problemdata/

UserChangeView http://localhost:8000/problemtag/

UserChangeAllView http://localhost:8000/uploadfile/

UserRegisterAPIView http://localhost:8000/problemtag/

UserLoginAPIView http://localhost:8000/uploadfile/

UserLogoutAPIView http://localhost:8000/problemtag/

UserUpdateRatingAPIView http://localhost:8000/uploadfile/

UserLoginDataView http://localhost:8000/userlogindata/

Wiki

此模块主要用来实现Wiki相关的API

models.py

model 功能

Wiki 算法详情

MBCode 模板介绍

MBCodeDetail 模板详细代码

TrainningContest 试炼谷的内容

Wiki

属性 功能 类型 说明

username 用户名 CharField 发布人姓名

type 发布的算法 CharField 发布了哪一篇算法

value 具体内容 TextField

time 发表时间 DateTimeField auto\_now

group 所属分组 CharField 新添加的算法的分类，仅std=1时有效

std 是否是标准算法 IntegerField 0代表是，1代表新添加的算法

title 新添加的算法标题 CharField 仅std=1时有效

MBCode

属性 功能 类型 说明

username 用户名 CharField primary\_key

des 模板介绍 CharField

time 添加时间 DateTimeField auto\_now

MBCodeDetail

属性 功能 类型 说明

username 用户名 CharField 发布人姓名

title 模板标题 CharField

des 模板介绍 DateTimeField

group 模板所属分组 CharField

code 模板代码 CharField

time 添加时间 DateTimeField auto\_now

TrainningContest

属性 功能 类型 说明

title 试炼名称 CharField

des 试炼介绍 CharField

tips 试炼提示 DateTimeField 对应Wiki中的type，用竖线隔开

group 试炼分组 CharField 第几章

num 试炼章节 CharField 第几关

problem 试炼题目 DateTimeField 中间用竖线隔开

permission.py

permission 读权限 写权限

WikiUserOnly 有 仅用户或管理员

UserOnly 有 仅用户或管理员

ManagerOnly 有 仅管理员

serializers.py

serializers 序列化字段

WikiSerializer ALL

WikiCountSerializer 不包括内容

MBCodeSerializer ALL

MBCodeDetailSerializer ALL

MBCodeDetailNoCodeSerializer 不包括代码

TrainningContestSerializer ALL

views.py

视图 查询集合 可过滤字段 权限 可分页 可搜索

WikiView ALL (‘username’, ‘type’, ‘group’, ‘std’,) WikiUserOnly 否 否

WikiCountView ALL (‘username’, ‘type’) UserOnly 否 否

MBCodeView ALL (‘username’,) UserOnly 否 否

MBCodeDetailView ALL (‘username’, ‘group’, ‘des’, ‘title’) UserOnly 否 否

MBCodeDetailNoCodeView ALL (‘username’, ‘group’, ‘des’, ‘title’) UserOnly 否 否

TrainningContestView ALL (‘group’, ‘title’,) ManagerOnly 否 否

urls.py

视图 访问路由

WikiView http://localhost:8000/wiki/

WikiCountView http://localhost:8000/wikicount/

MBCodeView http://localhost:8000/mbcode/

MBCodeDetailView http://localhost:8000/mbcodedetail/

MBCodeDetailNoCodeView http://localhost:8000/mbcodedetailnocode/

TrainningContestView http://localhost:8000/trainning/

修改后端

已存在的接口可以根据需要修改，这里将以添加一个模块为例，展示如何添加一个自己的后台模块或者在已有模块中添加一个API

在已有模块中添加一个API

现在我们假定要在Blog模块中添加一个博客回复功能

添加你的model

修改models.py文件，在文件最后添加

class BlogComment(models.Model):

# 此处编写你的字段

username = models.CharField(max\_length=50)

msg = models.CharField(max\_length=1000)

time = models.DateField(auto\_now=True)

# 复制粘贴即可

objects = models.Manager()

添加你的序列化器

修改serializers.py文件来

from .models import BlogComment # 记得import

class BlogCommentSerializer(serializers.ModelSerializer):

class Meta:

model = BlogComment # 设置为你的model

fields = '\_\_all\_\_'

具体的修改参考Django教程，这里使用REST Framework模块，使得这一切都如此简单

添加视图类

修改views.py来添加

from .models import BlogComment

from .serializers import BlogCommentSerializer

class BlogCommentView(viewsets.ModelViewSet): # 得益于rest\_framework，构建一个视图变得很方便

queryset = BlogComment.objects.all() # 查询集合，具体功能和修改，参考Django文档

serializer\_class = BlogCommentSerializer # 你的序列化器

filter\_fields = ('username', 'time') # 过滤字段，使得你可以在http请求中过滤，如 /?username=123456

pagination\_class = LimitOffsetPagination # 分页器，使得你的http请求支持分页，具体自行百度

permission\_classes = (ManagerOnly,) # 权限过滤器，具体参考Django文档

throttle\_scope = "post" # 限流类，直接复制粘贴即可

throttle\_classes = [ScopedRateThrottle, ]

添加路由信息

我们修改urls.py文件在 urlpatterns = [url(’’, include(routers.urls)),] 上方添加

routers.register('blogcomment', views.BlogCommentView) # 第一个参数为你的路由，第二个参数为你的视图

1

至此，我们成功的添加了一个自己的API

现在只需要同步你的数据库

python manager.py makemigrations blog

python manager.py migrate blog

1

2

现在可以通过\*\*http://localhost:8000/blogcomment/\*\*来访问你的API了

我们可以通过POST请求来添加一个数据，用PUT请求来修改一个数据

用DELETE请求来删除一个数据，用GET请求来获取所有数据

具体自行阅读HTTP请求的相关教程。

我们可以通过primary\_key来获取单个数据，如

http://localhost:8000/blogcomment/1/

Django会默认的生成一个ID字段作为primary\_key，当然你也可以自己制定，具体自行参阅Django教程。

我们可以通过limit和offset来实现分页，如

http://localhost:8000/blogcomment/?limit=50&offset=10

我们可以直接过滤设定好的字段，如

http://localhost:8000/blogcomment/?username=123&time=2019-5-29

这样一个普通的API很不安全，因此要做权限认证，自行编辑一个权限类，然后添加到permission\_classes中，具体代码编写自行参考Django教程

新建一个模块

我们可以直接新建一个模块，这非常简单，首先我们先执行如下命令

python manage.py startapp yourappname

1

这样就成功新建一个模块了，我们查看目录会发现多了一个yourappname的文件夹

然后我们要注册我们的模块，我们先修改Backend/Backend/setting.py文件

在INSTALLED\_APPS中添加你的模块名字

然后我们就可以参阅上面的在已有模块中添加一个API的教程来添加你的API

如果文件不存在，自行新建对应的文件在目录里即可。

我们添加完毕后要在Backend中注册你的路由信息。

我们修改Backend/Backend/urls.py在urlpatterns中添加

url(r’’, include(‘yourappname.urls’))

这样就可以了。

不要忘了同步你的数据库

python manager.py makemigrations yourappname

python manager.py migrate yourappname

1

2

以上方法是最简单的方法，大家学习Django后，会对添加API接口的方式更加熟悉，可以写得更加底层更加优雅~

前端开发

Vue.js 介绍

Vue.js是一个构建 Web 界面的成熟的渐进式框架。它的目标是通过尽可能简单的接口来实现响应式的数据绑定和组合的视图组件，然后再将这些组件组合起来。它不仅上手容易，而且还便于与第三方库或既有项目整合。本系统将采用它进行开发，而且学习起来非常简单，拓展性非常好，用户可以使用自己的组件，非常简单且高效的自定义自己的页面。 同时结合Element的UI库进行精致的页面设计，给用户呈现一个较好的网页。同时使用Js中的Axios库，可以异步的向后台请求数据，然后使用Js解析后台返回的Json数据，通过Vue的数据绑定功能，可以非常简单且便利的将数据呈现给用户。

系统结构

Vue.js 是使用 MVVC模式开发，所谓MVVC即 Model,View, Viewmodel。其中Model层用于存储数据，ViewModel层用于网页元素的变化和实现数据之间的双向绑定。View层用于显示数据。相比于其他框架，我们可以花费更多的代码时间在View和Model层的编写上，从而不必关心中间的消息是如何传递的。因为Vue在底层通过观察者模式已经很好地帮我们实现了。因此本系统主要通过如下方式实现数据绑定：

对于每一个页面都以组件的形式开发，然后组件与组件之间互不干扰。页面之间的跳转通过路由实现。组件内的数据在Created函数中通过Axois库进行后台的API访问，获取后绑定到data中，然后再由vue的model数据绑定，自动的呈现给用户，在之前我们只需要定义好页面即可。前端的页面采用Element库开发。

工具与框架介绍

Nodejs 一个让JavaScript运行在服务端的开发平台

NPM JavaScript的包管理工具

Webpack Webpack是一个模块打包器。主要目标是将JS等各种文件打包在一起,打包后的文件用于在浏览器中使用

Vue.js 一个前端框架

ElementUI 一个用Vue实现的前端库

Axios 一个Js实现的用于发送http请求的库

还有各种各样用Vue实现的前端库

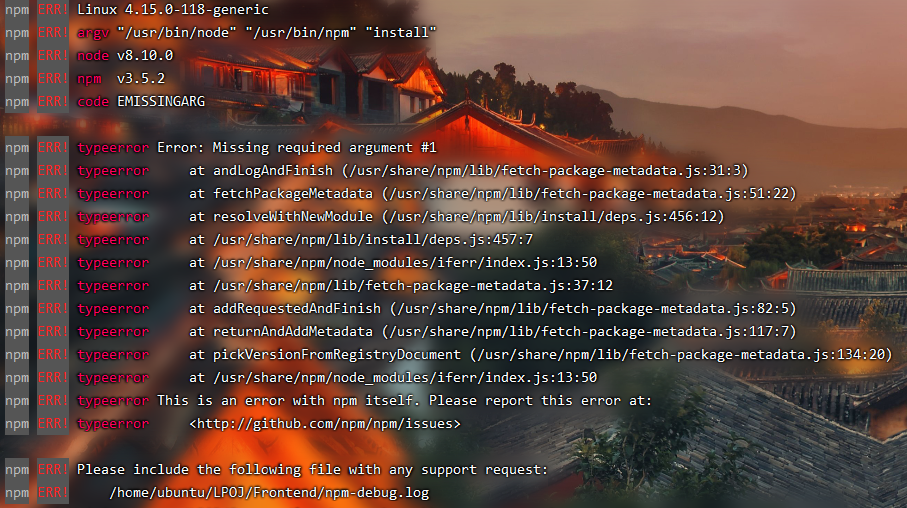
安装LPOJ前端

在开始开发前，让我们先做一些准备工作！

首先要安装nodejs和npm

不同系统根据自己的情况来安装，如果是Ubuntu系统，执行如下命令即可。

npm install出现问题



typeerror Error: Missing required argument #1

npm ERR! typeerror at andLogAndFinish (/usr/share/npm/lib/fetch-package-metadata.js:31:3)

npm ERR! typeerror at fetchPackageMetadata (/usr/share/npm/lib/fetch-package-metadata.js:51:22)

npm ERR! typeerror at resolveWithNewModule (/usr/share/npm/lib/install/deps.js:456:12)

npm ERR! typeerror at /usr/share/npm/lib/install/deps.js:457:7

npm ERR! typeerror at /usr/share/npm/node\_modules/iferr/index.js:13:50

npm ERR! typeerror at /usr/share/npm/lib/fetch-package-metadata.js:37:12

npm ERR! typeerror at addRequestedAndFinish (/usr/share/npm/lib/fetch-package-metadata.js:82:5)

npm ERR! typeerror at returnAndAddMetadata (/usr/share/npm/lib/fetch-package-metadata.js:117:7)

npm ERR! typeerror at pickVersionFromRegistryDocument (/usr/share/npm/lib/fetch-package-metadata.js:134:20)

npm ERR! typeerror at /usr/share/npm/node\_modules/iferr/index.js:13:50

npm ERR! typeerror This is an error with npm itself. Please report this error at:

npm ERR! typeerror <http://github.com/npm/npm/issues>

解决方法：使用淘宝镜像

sudo npm install -g cnpm --registry=https://registry.npm.taobao.org

sudo npm install

npm run-script build

npm run dev