新经资讯项目

一、项目分析

（一）新经资讯

1、新闻展示的Web项目

2、以抓取其他网站数据为新闻来源

3、基于Flask框架，前后端不分离

（二）技术实现

1、使用Flask框架实现

2、使用Redis + MySQL 进行数据存储

3、使用第三方扩展

（1）云通信

（2）七牛云

（三）功能模块分类

1、新闻模块

2、用户模块

3、后台管理模块

1. 项目目录说明

1、项目根目录 说明

/info 项目应用核心目录

/logs 项目日志目录

config.py 项目配置文件--保存session信息、调试模式、密钥等

manage.py 项目启动文件

requirements.txt 项目依赖文件

2、项目/info目录 说明

/libs 项目用到的资源库--第三方扩展(云通信)

/modules 项目模块--所有的蓝图对象和视图函数

/static 项目静态文件夹

/template 项目模板文件夹

/utils 项目通用设施--自定义状态码、上传图片等

\_\_init\_\_.py 项目应用初始化文件--应用程序实例、数据库实例、注册蓝图、日志等

constants.py 项目常量信息--数据库缓存信息、验证码、房屋信息等

models.py 项目模型类

3、项目/info/libs目录 说明

/yuntongxun 第三方扩展--发送短信

sms.py 发送短信

4、项目/info/static目录 说明

/admin/ 项目admin模块的静态文件，css/html/js等

/news/ 项目admin模块的静态文件，css/html/js等

favicon.ico 项目logo

5、项目/info/utils目录 说明

image\_storage.py 云存储设施文件--七牛云

response\_code.py 自定义状态码

注意：

一、用工厂方法来实例化应用对象app

def create\_app(config\_name):

app = Flask(\_\_name\_\_) //实例化启动app

app.config.from\_object(config[config\_name]) //导入配置信息并动态传入配置信息

db.init\_app(app) //关联db和app

Session(app) //把Session对象和app关联

CSRFProtect(app) // csrf保护关联app

\*\*此处使用请求钩子,在每次请求之后将csrf\_token的值传给前端页面\*\*

\*\*此处用来注册蓝图对象

from pro\_info.modules.news import news\_blue

app.register\_blueprint(news\_blue)

……

\*\*注册模板使用的过滤器\*\*

from pro\_info.utils.common import do\_index\_class

app.add\_template\_filter(do\_index\_class, 'index\_class')

return app

二、每次的请求都需要设置跨站请求保护，该方法可以使用请求钩子的方法，即每次请求之后执行,这个功能可以设置在实例化应用对象的时候就定义，即放在\_\_init\_\_.py文件中的实例化app对象的工厂方法中

@app.after\_request

def after\_request(response):

csrf\_token = csrf.generate\_csrf()

response.set\_cookie('csrf\_token',csrf\_token)

return response

并在前端的ajax中设置请求头

headers:{

X-CSRFToken:getCookie('csrf\_token')

}

三、由于很多接口需要判断用户是否在线的情况，对此我们采用的是用装饰器的方式实现这一功能的。

def login\_required(f): 🡺 定义一个方法，方便被调用

@functools.wraps(f) 🡺 这是一个python内置的装饰器工具，目的是让被装饰的函数的属性不会被改变

def wrapper(\*args,\*\*kwargs):

user\_id = session.get('user\_id') 🡺 尝试获取用户的登录信息

user = None

if user\_id:

try:

user = User.query.get(user\_id)

except Exception as e:

current\_app.logger.error(e)

g.user = user 🡺 使用引用上下文的g变量来保存用户的信息

return f(\*args,\*\*kwargs)

//具体的实现方式是让被装饰的函数的名称在返回wrapper之前赋值给wrapper的\_\_name\_\_ 🡺 wrapper.\_\_name\_\_ = f.\_\_name\_\_

return wrapper

新闻首页模块

一、注册相关接口

（一）图片验证码

1、获取前端生成的UUID编码

image\_code\_id = generate()前端调用该方法生成UUID编号，并发送给服务器

由于这是一个imp标签所以服务器可以request.args.get()获取到这个编码

2、调用captcha(图灵测试)扩展生成图片验证码

对获取到的参数进行验证，判断是否存在

存在则：name,text,image = captcha.generate\_captcha()调用captcha生成图片验证码

3、以前端获取的UUID为键，captcha生成的text为值，存储到Redis数据库中

使用Redis数据库redis\_store.setex(imageCodeId,time,text)将数据进行保存

4、使用flask中的make\_response将图片返回给前端页面

response = make\_response(image)

5、修改前端相应报头并返回

response. headers['Content-Type'] = 'image/jpg'

return….

（二）发送手机短信

1、根据接口文档指定请求方式，以及确定需要接受的参数

2、对接受的参数进行校验

（1）确认参数的完整性

if not all([mobile,image\_code,image\_code\_id])

return…

（2）确认手机号是否符合规范（采用正则的方式验证）

if not re.match(r‘1[3456789]\d{9}’,mobile)

return…

（3）确认验证码是否正确，从Redis数据库中取出之前保存的text值进行比对，为防止二次使用验证码，取出后删除数据库中数据

real\_image\_code = redis\_store.get(image\_code\_id)

redis\_store. delete(…)

if not real\_image\_code.lower() != image\_code.lower()

return…

3、参数校验完成后判断用户是否已经注册

查询MySQL数据库中是否存在该用户

user = User.query.filter\_by(mobile=mobile).first()

if user 表示用户存在已经注册

return…

4、生成6位随机数，并以用户手机号位键，随机数位值将数据存储到Redis数据库

sms\_code = '%06d' % random.randint(0,999999)

redis\_store.setex(mobile,time,sms\_code)

5、调用第三方云通讯接口发送生成的6位随机数给用户手机，

生成6位随机数：sms\_code = '%06d' % random.randint(0,999999)

调用云通讯发送短信：ccp = sms.CCP()

result = ccp.send\_template\_sms(mobile,[sms\_code,time / 60],1)

如果result == 0 表示发送成功，else发送失败

（三）注册按钮

1、根据接口文档指定请求方式，以及确定需要接受的参数

2、对接受的参数进行校验

（1）确认参数的完整性

if not all([mobile,sms\_code,password])

return…

（2）确认手机号是否符合规范（采用正则的方式验证）

if not re.match(r‘1[3456789]\d{9}’,mobile)

return…

（3）确认手机验证码是否正确，从Redis数据库中取出之前保存的sms\_code值进行比对，为防止二次使用，取出后删除数据库中数据

real\_sms\_code = redis\_store.get(mobile)

由于之前设置的是有时效的数据，因此需要判断是否还存在该sms\_code

redis\_store.delete(…)

if not real\_sms\_code != str(sms\_code)

return…

3、将用户的数据存储到MySQL数据库

user = User()

user.moblie = mobile

user.password = password(加密存储)

user.nike\_name = mobile(给用户默认设置一个昵称)

db.session.add(user)

db.session,commit()

4、为实现状态保持，将用户的注册的信息存储到Reids数据库中,并提示注册完成

session['user\_id'] = user.id

session['mobile'] = user.mobile

session['nick\_name'] = mobile

二、登录相关接口

1、根据接口文档确定需要接受的参数以及请求方式

2、校验参数的完整性

if not all([mobile,password])

3、判断手机号码是否符合规则

if not re.match(r‘1[3456789]\d{9}’,mobile)

return…

4、根据手机号码进行MySQL的数据查找

user = User.query.filter\_by(mobile=mobile).first()

5、判断是否存在用户或则密码是否正确

if user is None or not user.check\_password(password)（调用加密密码的检查匹对方法）

return

6、保存用户的登录信息并记录用户的最后登录时间

session['user\_id'] = user.id

session['mobile'] = mobile

session['nick\_name'] = user.nick\_name

user.last\_login = datetime.now()

7、记录的用户登录时间需要提交到数据库中

db.session.commit()

8、返回提示用户登录成功

三、退出相关接口

退出即删除用户的登录信息

session.pop('user\_id')

session.pop('mobile')

session.pop('nick\_name')

此处建议使用pop()方法，不建议使用clear()方法

四、主页相关接口

采用模板的方式，所以需要导入render\_template

1、确认用户是否登录在线

user = g.user //此处是定义的装饰器

2、展示点击排行按点击量进行排序查找，且根据前端的设计是显示的六条信息

news\_list = News.query.order\_by(News.clicks.desc()).limit(6)

3、判断点击排行数据查找结果是否存在

if not news\_list:

return…

4、定义一个列表容器保存查询结果

news\_dict\_list = []

5、遍历所有的查询对象并添加到列表容器中，并调用模型类中to.dict()的方法将之转换成字符串

for news in news\_list:

news\_dict\_list.append(news.to\_dict())

6、展示新闻分类数据需搜索所有分类信息

categories = Category.query.all()

7、判断分类数据是否存在

if not categories:

return

8、定义一个列表容器保存查询结果

category\_list = []

9、遍历所有的查询对象并添加到列表容器中，并调用模型类中to.dict()的方法将之转换成字符串

for category in categories:

category\_list.append(category.to\_dict())

10、定义一个字典保存所有数据，并将之传给模板进行数据填充

data = {

"user":user.to\_dict() if user else None,//表示如果用户未登录的情况下也是可以访问主页面的

"news\_dict\_list":news\_dict\_list,

"category\_list":category\_list

}

return render\_template(‘模板’，data=data)

五、首页新闻数据列表接口

1、根据接口文档定义路由，请求方式以及需要哪些参数等

2、接受所需要参数并进行校验（cid，page，per\_page）

为实现友好的交互，即使后端没传过来数据，在首页也是需要展示新闻的，所以我们默认会给其定义一个参数

cid = request.args.get('cid','1') 🡺分类id

page = request.args.get('page','1')

per\_page = request.args.get('per\_page','10')

3、校验参数，并将数字参数转换成整型值（为了和数据库进行查找）

try:

cid,page,per\_page = int(cid),int(page),int(per\_page)

except Exception as e:

return

4、根据参数进行查询数据库

（1）定义变量，存储过滤条件🡺根据分类id

filters = []

if cid > 1: => filters.append(News.category\_id == cid)

（2）默认按照分类id进行过滤，按新闻发布时间进行排序，对查询数据进行分页

paginate = News.query.filter(\*filters).order\_by(News.create\_time.desc()).paginate(page,per\_page,False)

paginate（页数，每页多少条目数，分页异常不报错）

5、获取分页后的新闻数据模型对象，总页数以及当前页数

news\_list = paginate.items //模型对象

total\_page = paginate.pages //总页数

current\_page = paginate.page //当前页数

6、遍历模型对象并添加到一个列表容器，使用to\_dict()方法转换成字符串

for news in news\_list:

news\_dict\_list.append(news.to\_dict())

7、将数据返回给模板页面进行填充

新闻详情页模块

一、详情页模板接口

1、获取参数以url传参的形式获取参数

@蓝图对象.route(‘/<int:news\_id>’)

def get\_news\_detail(news\_id):

2、判断用户是否登录在线，并获取用户信息

user = g.user

3、根据新闻id进行查询数据库

news = News.query.get(news\_id)

4、检查查询的结果

if not news：

return…

5、展示详情页的分类数据，查询数据库

categories = Category.query.all()

6、检查查询的结果，并定义容器保存遍历的查询对象

if not categories：

return…

7、将新闻详情的点击次数 +1

news.clicks += 1

8、判断是否收藏，默认情况为False，如果用户已经登录，并且被该用户收藏

is\_collected = False

if user and news in user.collection\_news:

is\_collected = True

9、评论内容，查询数据库获取当前新闻的评论情况

comments = []

comments = Comment.query.filter(Comment.news\_id == news\_id).order\_by(Comment.create\_time.desc()).all()

10、如果用户登录，获取当前登录用户的所有评论id

if user:

comment\_ids = [comment.id for comment in comments]

11、再查询点赞了哪些评论

comment\_likes = CommentLike.query.filter(CommentLike.comment\_id.in\_(comment\_ids),CommentLike.user\_id == g.user.id).all()

12、获取点赞的评论数据

comment\_like\_ids = [comment\_like.comment\_id for comment\_like in comment\_likes]

13、定义一个列表用户来保存评论数据

comment\_dict\_li = []

14、遍历查询到的点赞的评论数据

for comment in comments:

comment\_dict = comment.to\_dict()

15、判断评论是否被点赞，默认为False

comment\_dict['is\_like'] = False

if comment.id in comment\_like\_ids:

comment\_dict['is\_list'] = True

16、将查询到的点赞的评论数据保存到之前定义的列表中

comment\_dict\_li.append(comment\_dict)

17、如果用户是登录的，查询用户关注的新闻发布者，默认为Falas

is\_followed = False

if news.user and user:

if news.user in user.followers:

is\_followed = True

18、将所有数据添加到一个字典容器中

data = {

"news":news.to\_dict(),

"category\_list":category\_list,

'user': user.to\_dict() if user else None,//如果用户未登录，那么有关用户的信息就为False，新闻详情页没必要规定用户必须在线

'is\_collected': is\_collected,

'is\_followed': is\_followed,

'comments': comment\_dict\_li

}

19、调用模板页面返回数据，并对模板进行数据填充显示

return render\_template('news/detail.html',data=data)

二、收藏和取消收藏接口

根据接口文档进行路由分析，指定请求方式

1、尝试获取用户信息，如果用户未登录，需提示用户登录，才能进行收藏

user = g.user

if not user:

return…

2、获取参数news\_id，action（collect，cancle\_collect）

request.json.get()方法获取

3、检查参数，对news\_id强转为整型值，对数据库进行查询，如果强转出错返回错误信息

4、检查action的值是否存在

if action not in ['collect', 'cancel\_collect']:

return…

5、判断新闻是否存在数据库中

news = News.query.get(news\_id)

if not news:

return…

6、判断选择的是collect还是cancle\_collect,并判断用户是否收藏过该新闻,未收藏过添加到收藏，收藏过返回已收藏

if action == collect:

if news not in user.collection\_news:

user.collection\_news.append(news)

return…

return…

7、返回提示信息

三、关注和取消关注接口

与收藏接口类似，只需根据接口文档定好路由以及请求方式，不过多赘述

四、新闻评论接口

根据接口文档确定路由及请求方式

1、获取用户登录信息，如果用户未登录直接返回并提示登录

user = g.user

2、获取参数，news\_id,comment,parent\_id(回复的评论的id)

request.json.get()方法

3、校验参数完整性

if not all([news\_id,comment]):

retuan…

4、对news\_id进行强转，并判断是否有parent\_id，如果强转失败返回错误信息

news\_id = int（news\_id）

if parent\_id:

parent\_id = int（parent\_id）

5、查询新闻并判断新闻是否存在

news = News.query.get(news\_id)

if not news：

return…

6、实例化评论模型的对象，并添加数据

comment = Comment()

comment.user\_id = user.id

comment.news\_id = news.id

comment.content = comment\_conent

7、判断父评论是否存在，存在则添加父评论的信息

if parent\_id：

comment.parent\_id = parent\_id

8、提交数据到数据库

try:

db.session.add(comment)

db.session.commit()

except Exception as e:

current\_app.logger.error(e)

db.session.rollback()

9、返回评论数据给前端ajax

五、点赞和取消点赞接口

根据接口文档确定路由及请求方式

1、获取用户登录信息，如果用户未登录直接返回并提示登录

user = g.user

2、获取参数，comment\_id,action(add,remove)

request.json.get()方法

3、校验参数完整性

if not all([comment\_id,action])

return…

if action not in [add,remove]

return…

4、把comment\_id转成整型，强转错返回错误信息

5、根据comment\_id查询数据

comment = Comment.query.get(comment\_id)

6、判断查询结果，如果不存在，返回错误信息

7、判断action参数是否为add或者remove

（1）如果为add，判断行为是点赞还是取消点赞，使用CommentLike进行过滤查询（user\_id，comment.id）

comment\_like\_model = CommentLike.query.filter(CommentLike.user\_id == user.id, CommentLike.comment\_id == comment.id).first()

if not comment\_like\_model：🡺 点赞行为，点赞数加1

comment\_like\_model = CommentLike()

comment\_like\_model.user\_id = user.id

comment\_like\_model.comment\_id = comment.id

db.session.add(comment\_like\_model)

comment.like\_count += 1

（2）否则为取消点赞 🡺 取消点赞，点赞数减1

db.session.delete(comment\_like\_model)

comment.like\_count -= 1

8、将数据提交到数据库中

db.session.commit()

9、返回结果给ajax

个人中心模块

基于iframe进行实现，子页面的数据更新之后需要同步主页面相关联数据，可以采用js进行实现，本项目即采用了ajax的数据交互方式

一、用户页面接口

这里我们用到的是之前就定义好的装饰器来获取的用户信息

1、获取用户信息

user = g.user

2、判断是否获得了用户的信息，如果没有则重定向到主页

if not user:

return redirect(‘/’)

3、将获取的用户信息使用字典容器保存起来

4、将数据返回给user.html模板，进行页面显示

二、用户基本信息展示和修改接口

根据接口文档确定路由以及请求方式[‘GET’,‘POST’]

1、判断请求方式，默认GET请求加载模板页面，可以直接以装饰器获取到用户的信息进行返回

user = g.user

if request.method == ‘GET’:

将用户信息返回给模板

return… 🡺 模板为：user\_base\_info.html

2、获取参数（signature,nick\_name,gender(男，女)）

request.json.get()方法

3、校验参数的完整性

if not all([signature,nick\_name,gender])

return…

4、检查参数gender参数

if gender not in [‘MAN’，‘WOMEN’]

return…

5、使用user对象将用户的基本信息保存到数据库

user.signature = signature

user.nick\_name = nick\_name

user.gender = gender

try:

db.session.add(user)

db.session.commit()

except Exception as e:

current\_app.logger.error(e)

db.session.rollback()

6、返回结果

三、上传用户头像接口

根据接口文档确定路由以及请求方式[‘GET’,‘POST’]

1、判断请求方式，默认GET请求加载模板页面，可以直接以装饰器获取到用户的信息进行返回

user = g.user

if request.method == ‘GET’:

将用户信息返回给模板

return… 🡺 模板为：user\_pic\_info.html

2、获取参数（avatar）

request.files.get()方法

3、校验参数是否存在

if not avatar：

return…

4、使用read()方法读取图片的二进制数据并保存

avatar\_data = avatar.read()

5、调用第三方接口storage()，将图片上传到七牛云，并保存七牛云返回的图片名称

image\_name = storage(avatar\_data)

6、给用户保存图片的名称并提交到到数据库

user.avatar\_url = image\_name

try:

db.session.add(user)

db.session.commit()

except Exception as e:

current\_app.logger.error(e)

db.session.rollback()

7、拼接图片的绝对路径并返回给ajax

设置的七牛云地址 + image\_name

四、修改用户密码接口

根据接口文档确定路由以及请求方式[‘GET’,‘POST’]

1、判断请求方式，默认GET请求加载模板页面

if request.method == ‘GET’:

return… 🡺 模板为：user\_pass\_info.html

2、获取参数（old\_password,new\_password）

request.json.get()方法

3、检查参数完整性

if not all([old\_password,new\_password])

return…

4、获取用户信息，并比较用户输入的旧密码，验证是否正确

user = g.user

if not user.check\_password(old\_password):

return…

5、验证成功后保存用户的新密码，并提交到数据库

user.password = new\_password

try:

db.session.add(user)

db.session.commit()

except Exception as e:

current\_app.logger.error(e)

db.session.rollback()

6、返回结果给ajax

五、新闻发布接口

根据接口文档确定路由以及请求方式[‘GET’,‘POST’]

1、判断请求方式，默认GET请求加载模板页面

if request.method == 'GET':

查询新闻分类，与新闻首页模块的主页相关接口6、7、8、9雷同

return… 🡺 模板为：user\_ news\_release.html

2、获取请求参数

request.form.get()方法获取表单参数

request.files.get()方法获取新闻图片参数

3、校验参数完整性

if not all([title,category\_id,digest,index\_image,content]):

return…

4、强转分类id，如果报错直接返回错误信息

5、read()方法读取获取的新闻图片的二进制数据并调用第三方七牛云接口上传图片，并保存返回的图片名称

image\_data = index\_image.read()

image\_name = storage(image\_data)

6、实例化新闻类对象，用来保存新闻数据

news = News() 🡺 新闻类对象

news.title = title 🡺 新闻标题

news.source = '个人发布' 🡺 新闻发布机构

news.category\_id = category\_id 🡺 新闻分类id

news.digest = digest 🡺 新闻摘要

news.index\_image\_url = 设置的七牛云地址 + image\_name 🡺 新闻图片绝对路径

news.content = content 🡺 新闻内容

news.status = 1 🡺 新闻状态，1表示待审核，0表示审核通过，-1表示未通过审核

7、提交数据到数据库中

六、用户新闻列表接口

根据接口文档确定路由和请求方式

1、获取参数，页数，默认1

request.args.get()方法

2、校验参数，将page强转为整型，如果报错直接返回错误信息

3、获取用户信息，定义容器存储查询结果，总页数默认1，当前页默认1

4、查询数据库，查询新闻数据并进行分页，

user = g.user

paginate = News.query.filter(News.user\_id==user.id).paginate(page,每页数据数目 ,False)

\*\*可参看新闻首页模块的新闻列表接口\*\*

5、返回数据给模板：user\_news\_list.html

七、用户关注信息接口

根据接口文档确定路由和请求方式

\*\*与用户新闻列别接口流程类似\*\*

返回数据给模板：user\_follow.html

八、查询用户关注的其他用户信息

根据接口文档确定路由和请求方式

1、获取用户的登录信息

user = g.user

2、获取参数other\_id(用户关注的用户)

request.args.get()方法

3、校验参数是否存在，如果不存在，返回错误信息

4、查询信息

other = User.query.get(other\_id)

5、判断新闻是否有作者，且用户关注过作者，默认为False

is\_follwed = False

if other and user:

if other in user.followed:

is\_follwed = True

6、返回数据给模板：other.html

九、返回指定用户发布的新闻接口

根据接口文档确定路由和请求方式

1、获取参数(user\_id,页数默认为1)

request.args.get()方法获取

2、校验参数，强转页数的参数，如果报错，直接返回错误信息

3、判断用户是否存在

other = User.query.get(user\_id)

if not other:

return…

4、如果用户存在，分页用户发布的新闻数据

paginate = other.news\_list.paginate(page,每页条目数,False)

5、获取分页数据，并遍历数据

news\_list = paginate.items

current\_page = paginate.page

total\_page = paginate.pages

6、返回数据给ajax

后台管理模块