flask项目笔记

CMS后台

基本目录搭建

- apps
  - cms 后台
  - front 前台
  - common 共有的
- static
  - cms
  - front
- templates
  - cms
  - front
- bbs.py 程序的入口
- config.py 配置文件
- manage.py 数据库映射文件
- exts.py db文件

CMS登录页面

- 模板渲染
- 后台用户的注册
  - 通过命令行
  - -u juran -p 111111 -e jr@qq.com
- 完成登录功能
  - 表单验证
  - 表单错误信息的重写
- 登录的验证
  - 装饰器
  - 钩子函数
    - 是否是登录页面
    - 重定向
- CSRF保护
  - CSRFProtect 在入口文件中

    - ```
      CSRFProtect(app)
      ```
    - 在模板中发送post的请求的时候要加上 csrf字符串

CMS后台功能

cms首页

- 加载静态资源文件
- CMS用户名渲染
    - 钩子函数
        - 查询用户是否存在
        - 将用户信息绑定到g对象上
- 注销
    - 删除 session中的user_id
    - 重定向到登录页面
- 基类模板的定义
- 个人页面

cms后台修改密码

- form表单验证
    - 密码长度
    - 两次密码输入是否一致
    - 表单的继承 错误方法
- ajax完成修改密码
- ```
  提交的方式  post  get
  地址       url
  数据       data
  success    成功
  fail       失败
  ```
- 完善josn返回数据
    - ```
      单独定义一个 工具文件  restful.py
      ```
- JS提示框
    - sweetalert

cms后台发送邮箱
      ```

- 邮箱的模板渲染
- ```
  发送邮件   pip install Flask-Mail
  ```

权限管理

二进制

```
运算
&  与
|  或
用户b 有么有a权限
角色的权限 & 用户b的权限  = 用户b的权限
```

1589810921653

用户 -> 角色 -> 权限

权限

访问者权限

管理帖子权限

管理评论

管理板块

管理前台用户

管理后台用户

管理后台管理员

角色有的权限

访问者角色：访问权限

运营者角色：访问，管理帖子，管理评论，管理后台用户，管理前台用户

管理员：访问，管理帖子，管理评论，管理后台用户，管理前台用户，管理板块

开发者：所有权限

角色模型的定义

```
通过中间表 来完成   用户和角色之前的关联
```
```

通过命令行的方式 来添加的用户角色

权限的判断

在用户模型中

```
@property
def permissions(self):
    # 判断用户是否有角色
    if not self.roles:
    return 0

    all_permissions = 0

    # 获取角色的所有的权限
    for role in self.roles:
    permissions = role.permissions
    all_permissions |= permissions
    return all_permissions

def has_permission(self, permission):
    all_permissions = self.permissions
    result = all_permissions & permission == permission
    return result
```

客户端的验证

```


@cms_bp.context_processor
def cms_context_processor():
    return {"CMSPersmission": CMSPersmission}

{% set user = g.cms_user %}
{% if user.has_permission(CMSPersmission.POSTER) %}
<li class="nav-group post-manage"><a href="#">帖子管理</a></li>
{% endif %}
```

服务端的验证

```
def permission_required(permission):
    def outter(func):
        @wraps(func)
        def inners(*args, **kwargs):
            user = g.cms_user
            if user.has_permission(permission):
                return func(*args, **kwargs)
            else:
                return redirect(url_for('cms.index'))
        return inners
    return outter
```

前台

写HTML？ 功能性的开发

用户模型创建

```
需求分析  保存那些字段

def __init__(self, *args, **kwargs):
    if "password" in kwargs:
    self.password = kwargs.get('password')
    kwargs.pop('password')

    # super(FrontUser, self).__init__(*args, **kwargs)
    super().__init__(*args, **kwargs)
```

添加前端用户

```
manage.py
```

注册页面完成

```
视图类->渲染模板
```

图片验证码

发送短信验证码

```
云片
阿里大于
聚合数据          发送短信    免费API
```

```
申请模板
申请签名
防止你恶意的发送短信

API 文档查看

整合到flask项目中


通过ajax去发送的请求

接口的加密  JS加密

云片发送短信  1小时/3

验证码保存到Redis里面
```

完成注册功能

- 表单提交
  - 写form表单验证数据
  - 蓝图里面引入表单
  - 功能-》保存数据到数据库中

登录功能

- 表单提交
  - 写form表单验证数据
  - 蓝图里面引入表单
  - 功能-》验证 手机号 密码

跳转回原页面

```
request.referrer    是页面跳转  不是url地址直接访问  直接访问 referrer none
```

首页搭建

通过bootcss搭建页面

```
导航条
轮播图
帖子
```

cms后台轮播图管理

上传轮播图

修改录播图

删除录播图

前台的轮播图显示

CMS后台板块管理

发表文章的板块

文章属于那个板块

Python面向对象->Python基础

七牛JS和Python的SDK使用

安装

```
pip install qiniu
```

编写获取uptoken的接口

```
import qiniu
@app.route('/uptoken/')
def uptoken():
    # AK
    access_key = "xxxxxxx"
    # SK
    secret_key = "xxxxxxx"
    # 验证
    q = qiniu.Auth(access_key,secret_key)
    # 储存空间名字
    bucket = "tokimeki"
    token = q.upload_token(bucket)
    return jsonify({"uptoken":token})
```

在前端中添加js的sdk:七牛为javascript提供了一个专门用来传文件的接口

```
<script src="https://cdn.staticfile.org/Plupload/2.1.1/moxie.js"></script>
<script src="https://cdn.staticfile.org/Plupload/2.1.1/plupload.dev.js"></script>
<script src="https://cdn.staticfile.org/qiniu-js-sdk/1.0.14-beta/qiniu.js"></script>
```

在前段添加zlqiniu.js，js文件封装了七牛的初始化和配置相关的参数

```
<script src{{ url_for('static',filename='zlqiniu.js') }}""></script>
```

初始化七牛:使用以下代码初始化七牛，配置一些参数

```
$(function () {
    lgqiniu.setUp({
        'domain': 'http://qaulohbyc.bkt.clouddn.com/',
        'browse_btn': 'upload-btn',
        'uptoken_url': '/c/uptoken/',
        'success': function (up,file,info) {
            var imageInput = $("input[name='image_url']");
            imageInput.val(file.name);
        }
    });
});
```

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <script src="https://cdn.staticfile.org/Plupload/2.1.1/moxie.js"></script>
    <script src="https://cdn.staticfile.org/Plupload/2.1.1/plupload.dev.js"></script>
    <script src="https://cdn.staticfile.org/qiniu-js-sdk/1.0.14-beta/qiniu.js"></script>
    <script src="{{ url_for('static',filename='lgqiniu.js') }}"></script>
    <script>
        window.onload = function () {
            lgqiniu.setUp({
                /*http://7xqenu.coml.z0.glb.clouddn.com/*/
                'domain': 'http://pdhjzz2pa.bkt.clouddn.com/',
                'browse_btn': 'upload-btn',
                'uptoken_url': '/uptoken/',
                'success': function (up,file,info) {
                    var image_url = file.name
                    var imageInput = document.getElementById("image-input");
                    imageInput.value = image_url;

                    var im = document.getElementById("img");
                    im.setAttribute("src",image_url);

                }
            });
        }
```

```
    </script>
</head>
<body>
    <button id="upload-btn">上传文件</button>
    <input type="text" id="image-input">
    <img src="" alt="" id="img"/>
</body>
</html>
```

前台帖子发布

models

```
 class PostModel(db.Model):
    __tablename__ = 'post'
    id = db.Column(db.Integer, primary_key=True, autoincrement=True)
    title = db.Column(db.String(200), nullable=False)
    content = db.Column(db.Text, nullable=False)
    content_html = db.Column(db.Text)
    create_time = db.Column(db.DateTime, default=datetime.now)
    board_id = db.Column(db.Integer, db.ForeignKey("board.id"))
    author_id = db.Column(db.String(100), db.ForeignKey("front_user.id"), nullable=False)

    board = db.relationship("BoardModel", backref="posts")
    author = db.relationship("FrontUser", backref='posts')

    @staticmethod
    def on_changed_content(target, value, oldvalue, initiator):
        allowed_tags = ['a', 'abbr', 'acronym', 'b', 'blockquote', 'code',
                        'em', 'i', 'li', 'ol', 'pre', 'strong', 'ul',
                        'h1', 'h2', 'h3', 'p', 'img','video', 'div', 'iframe', 'p', 'br',
'span', 'hr', 'src', 'class']
        allowed_attrs = {'*': ['class'],
                         'a': ['href', 'rel'],
                         'img': ['src', 'alt']}
        target.content_html = bleach.linkify(bleach.clean(
            markdown(value, output_format='html'),
            tags=allowed_tags, strip=True, attributes=allowed_attrs))


db.event.listen(PostModel.content, 'set', PostModel.on_changed_content)
```

发布帖子优化

- 加上验证码
    - 防止恶意灌水

celery

项目部署到阿里云

为什么会选择云服务器部署?

- 为了固定的IP
- 本地的IP是一个动态分配的IP地址,重启路由之后可能会变

用的比较多的就是腾讯云,与阿里云

阿里云安全组规则

端口范围

80/80

3306/3306

6379/6379

23/23

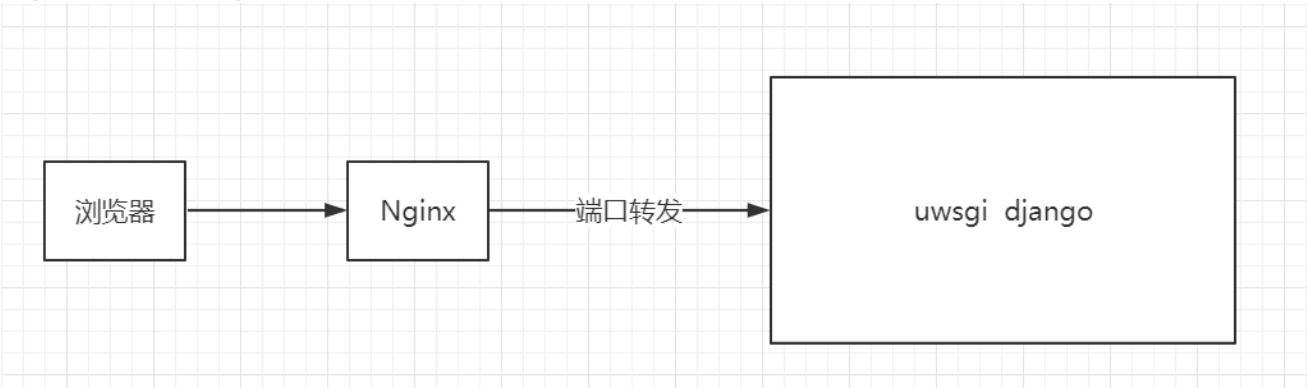
443/433

22/22

80/80

3389/3389

nginx + uwsgi



Nginx默认是80端口

uwsgi是C语言写的,拉起flask运行,flask本身的并发性不好

1.安装Python3.7(不是必须的)

1.安装依赖包

```
yum install opensll-devel bzip2-devel expat-devel gdbm-devel readline-devel sqlite-devel gcc
gcc-c++ opensll-devel libffi-devel python-devel mariadb-devel
```

2.下载Python源码

```
wget https://www.python.org/ftp/python/3.7.3/Python-3.7.3.tgz
tar -xzvf Python-3.7.3.tgz -C /tmp
cd /tmp/Python-3.7.3
```

3.把Python3.7安装到 /usr/local目录

```
./configure --prefix=/usr/local
make
make altinstall  # 这一步比较耗时
```

4.更改/usr/bin/python链接

```
ln -s /usr/local/bin/python3.7 /usr/bin/python3
ln -s /usr/local/bin/pip3.7 /usr/bin/pip3
```

2.MySQL和Redis

安装MySQL

1.下载 MySQL yum包

```
wget http://repo.mysql.com/mysql57-community-release-el7-10.noarch.rpm
```

2.安装MySQL源

```
rpm -Uvh mysql57-community-release-el7-10.noarch.rpm
```

3.安装MySQL服务端,需要等待一些时间

```
yum install -y mysql-community-server
```

4.启动MySQL

```
systemctl start mysqld.service
```

5.检查是否启动成功

```
systemctl status mysqld.service
```

6.获取临时密码，MySQL5.7为root用户随机生成了一个密码

```
grep 'temporary password' /var/log/mysqld.log
```

7.通过临时密码登录MySQL，进行修改密码操作

```
mysql -uroot -p
```

8.因为MySQL的密码规则需要很复杂，我们一般自己设置的不会设置成这样，所以我们全局修改一下

```
mysql> set global validate_password_policy=0;
mysql> set global validate_password_length=1;

# 修改密码
ALTER USER 'root'@'localhost' IDENTIFIED BY 'yourpassword';
```

9.授权其他机器远程登录

```
GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' IDENTIFIED BY 'yourpassword' WITH GRANT OPTION;

FLUSH PRIVILEGES;
```

10.设置MySQL的字符集为UTF-8，令其支持中文

```
vim /etc/my.cnf

[mysql]
default-character-set=utf8
```

11.重启MySQL

```
systemctl restart mysqld.service
```

12.查看MySQL运行状态

```
ps -aux|grep mysqld
```

3.安装Redis

1.安装redis

```
yum install redis

systemctl start redis
```

2.连接Redis

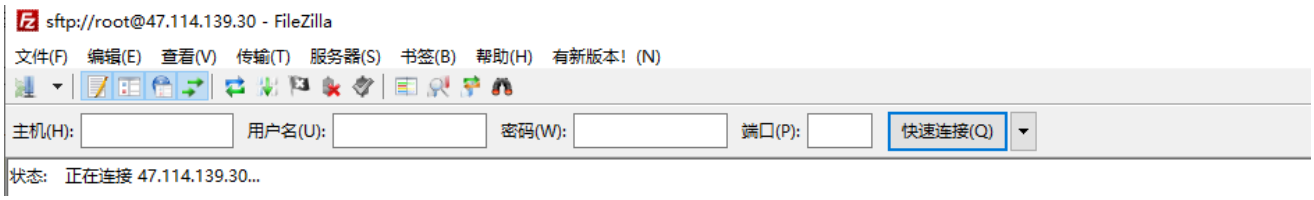
```
redis-cli
```

4.安装虚拟环境

```
pip3 install pipenv

mkdir flask-project
```

5.FileZilla上传项目



将本地的环境导出，以便在服务器上安装

```
pip freeze > requirements.txt
```

将本地环境安装到服务器上

```
pip install -r requirements.txt
```

创建数据库

```
create database bbs charset=utf8;
```

重新映射数据库

```
# 删除原有的映射文件
rm -rf migrations/

python manage.py db init

python manage.py db migrate

python manage.py db upgrade
```

映射完成后，通过ip去访问

```
http://47.114.139.30:5000/
```

6.安装uwsgi

uwsgi是一个应用服务器，非静态文件的网络请求就必须通过他完成， 他也可以充当静态文件服务器，但不是他的强项。uwsgi是使用python编写的，因此通过pip3 install uwsgi就可以了。(uwsgi必须安装在系统级别的Python环境中，不要安装到虚拟环境中)。然后创建一个叫做uwsgi.ini的配置文件：

```
[uwsgi]

# 必须全部为绝对路径
# 项目的路径
chdir           = /root/flask-project/bbs/
# flask的wsgi文件
wsgi-file       = /root/flask-project/bbs/bbs.py
# 回调的app对象
callable        = app
# Python虚拟环境的路径  pipenv --venv 进入到虚拟环境,目录里面执行
home            = /root/.local/share/virtualenvs/flask-project--bwy33Ao

# 进程相关的设置
# 主进程
master          = true
# 最大数量的工作进程
processes       = 10

http            = :5000

# 设置socket的权限
chmod-socket    = 666
# 退出的时候是否清理环境
vacuum          = true
```

退出虚拟环境： deactivate

依赖环境

```
yum install -y gcc* pcre-devel openssl-devel
```

然后通过命令uwsgi --ini uwsgi.ini运行，确保没有错误。然后在浏览器中访问http://ip地址:5000，如果能够访问到页面（可能没有静态文件）说明uwsgi配置没有问题。

安装和配置nginx

虽然uwsgi可以正常的部署我们的项目了。但我们还是依然要采用nginx来作为web服务器。使用nginx来作为web服务器有以下好处：

1. uwsgi对静态文件资源处理并不好，包括响应速度，缓存等。
2. nginx作为专业的web服务器，暴露在公网上会比uwsgi更加安全一点。
3. 运维起来更加方便。比如要将某些IP写入黑名单，nginx可以非常方便的写进去。而uwsgi可能还要写一大段代码才能实现。

安装

通过yum install nginx即可安装。

nginx简单操作命令

- 启动：systemctl start nginx
- 关闭：systemctl stop nginx
- 重启：systemctl restart nginx

添加配置文件

在/etc/nginx/conf.d目录下，新建一个文件，叫做bbs.conf，然后将以下代码粘贴进去：

```
upstream bbs{
    server 127.0.0.1:5000;
}

# 配置服务器
server {
    # 监听的端口号
    listen      80;
    # 域名
    server_name 47.114.139.30;
    charset     utf-8;

    # 最大的文件上传尺寸
    client_max_body_size 75M;

    # 静态文件访问的url
    location /static {
        # 静态文件地址
        alias /root/flask-project/bbs/static;
    }

    # 最后，发送所有非静态文件请求到flask服务器
    location / {
        uwsgi_pass  127.0.0.1:5000;
        # uwsgi_params文件地址
        include     /etc/nginx/uwsgi_params;
    }
}
```

写完配置文件后，为了测试配置文件是否设置成功，运行命令：service nginx configtest，如果不报错，说明成功。 每次修改完了配置文件，都要记得运行systemctl start nginx。