Django实践以及服务器部署

- Django的安装以及环境配置可以通过 Django入门与实践 来学习。
- Django入门指南

常见命令

- 数据库移植

```
python manage.py makemigrations
python manage.py migrate
```

- 创建超级管理员

```
python manage.py createsuperuser
```

- 启动服务器

```
python manage.py runserver(默认是8000端口)
```

- 导出数据

```
python manage.py dumpdata appname > appname.json
python manage.py loaddata appname.json
```

- 清空数据库

```
python manage.py flush
```

- Django项目环境终端

```
python manage.py shell
```

- 数据库命令行

```
python manage.py dbshell
```

- 创建新项目

```
django-admin startproject project_name
```

- 创建新App

```
python manage.py startapp app_name
```

项目结构

Painting

> mainapp
>
> > **init**.py
> >
> > migrations
> >
> > admin.py
> >
> > apps.py
> >
> > forms.py
> >
> > models.py
> >
> > serializers.py
> >
> > tests.py
> >
> > urls.py
> >
> > views.py
>
> media
>
> Painting
>
> > **init**.py
> >
> > settings.py
> >
> > urls.py
> >
> > wsgi.py
>
> static
>
> db.sqilte3
>
> manage.py
>
> requirements.txt

- Painting
  - settings.py：app配置文件。
  - urls.py：URL配置文件，配置页面URL。
  - wsgi.py：python服务器网关接口。
- manage.py：项目与命令行交互，项目管理器。
- db.sqlite3：数据库
- media：存储图片，音乐
- static：存储静态文件，比如.jsp，.html，.css。
- migrations：数据库移植模块，自动生成。
- mainapp

- admin.py：后台管理系统配置
- apps.py：
- models.py：数据库建立代码
- forms.py：表结构代码
- serializers.py：序列化数据
- tests.py：自动化测试模块
- urls.py：前端URL
- views.py：执行响应的逻辑代码

简单小例子

- 建立一个博客系统，需要进行一下代码编码。
  - `models.py`

```
class Blog(models.Model):
    BlogId = models.CharField(max_length=100, primary_key=True)  # Id
    BlogName = models.CharField(max_length=30)  # 博客名
    Author = models.CharField(max_length=30)  # 作者

    def __str__(self):
        return self.BlogId
```

  - `serializers.py`

```
class BlogSerializer(serializers.ModelSerializer):
    class Meta:
        model = Blog
        fields = '__all__'
```

  - `views.py`

```
class BlogSet(viewsets.ModelViewSet):
    queryset = Blog.objects.all()
    serializer_class = BlogSerializer
```

  - `myBlog/urls.py`

```
url(r'blog/$',views.BlogSet)
```

  - `settings.py`

```
INSTALLED_APPS = ['myBlog']
```

```
LANGUAGE_CODE 设置为'zh_hans'，也可以设置为'zh-Hans'，但是在部署的时候第二个会出错，建议第一个。
```

  - `blog/urls.py`

```
url(r'',include('myBlog.urls',namespqce='myBlog'))
```

- 以上代码就是建立一个博客系统，没有前端，可以通过`127.0.0.1:8000/blog`访问，返回`json`对象。
- 访问`static/media`文件
  - `settings.py`

    ```
    STATIC_URL = '/static/'
    MEDIA_URL = '/media/'
    MEDIA_ROOT = os.path.join(BASE_DIR,'media')
    STATIC_ROOT = os.path.join(BASE_DIR,'static')
    ```

  - `blog/urls.py`

    ```
    url(r'^static/(?P<path>.*)$', serve, {'document_root': settings.STATIC_ROOT}),
    url(r'^media/(?P<path>.*)$', serve, {'document_root': settings.MEDIA_ROOT}),
    ```

- 以上就是编写简单的博客系统，下面讲解一下将博客系统部署到云服务器上，生产上线。

使用Nginx和Gunicorn部署Django项目

- 满足条件
  - 可以通过外网访问的服务器
  - 域名(当然没有域名也可以，直接通过IP进行访问)

搭建服务器

- 本教程使用的本地环境是Windows10，服务器环境为Ubuntu 16.01（64位）。

安装软件

- 新的服务器的用户是root，我们需要新建一个拥有超级权限的新用户。

```
#在root下创建一个新用户，wangyufei是用户名
root@localhost:~# useradd -m -s /bin/bash wangyufei

#把新创建的用户加入到超级权限组
root@localhost:~# usermod -a -G sudo wangyufei

#为新用户设置密码
root@localhost:~# passwd wangyufei

#切换到新用户
root@localhost:~# su - wangyufei

#切换成功，注意到root已经变为wangyufei了
wangyuyfei@localhost:~$
```

- 如果是新服务器的话，需要更新一下系统。

```
wangyuyfei@localhost:~$ sudo apt-get update
wangyuyfei@localhost:~$ sudo apt-get upgrade
```

- 安装软件，用到的软件有`Nginx`、`Git`、`pip`、`Django`、`virtualenv`和`Anaconda`。

```
#安装Anaconda，找到最新的Anaconda版本
https://mirrors.tuna.tsinghua.edu.cn/anaconda/archive/

#下载
wangyuyfei@localhost:~$ wget
https://mirrors.tuna.tsinghua.edu.cn/anaconda/archive/Anaconda3-5.0.1-Linux-
x86_64.sh

#安装
wangyuyfei@localhost:~$ bash Anaconda3-5.0.1-Linux-x86_64.sh

#Linux里面默认的是python2.7，我们需要切换到python3.6
wangyuyfei@localhost:~$ echo 'export PATH="/home/hqy/anaconda2/bin:$PATH"' >>
~/.bashrc
wangyuyfei@localhost:~$ source ~/.bashrc
#检查一下，看是否是python3.6
wangyuyfei@localhost:~$ python --version

#安装Nginx
wangyuyfei@localhost:~$ sudo apt-get install nginx

#安装git
wangyuyfei@localhost:~$ sudo apt-get install git

#安装pip
wangyuyfei@localhost:~$ sudo apr-get install python3-pip

#安装virtualenv
wangyuyfei@localhost:~$ sudo pip install virtualenv(可能需要更新pip,使用命令更新)

#安装Django
sudo pip install django
```

解析域名

- 将域名和服务器的IP地址绑定后，可以通过域名访问服务器。

启动Nginx服务

- Nginx是用来处理静态文件请求的，比如说访问一个博客文章的页面时，服务器会收到两种请求：
  - 显示文章的详细信息，这些信息保存在数据库中
  - 图片、CSS、js等存在服务器某个文件夹下的静态文件。
- 前面我们已经安装了Nginx，并且域名已经和IP地址绑定了，运行下面的命令启动Nginx服务：

```
wangyuyfei@localhost:~$ sudo service nginx start(这里需要注意一下，服务器需要开放80端口，
外网才能进行访问)
```

在浏览器输入域名，看到如下页面说明Nginx启动成功了。

部署代码

部署前的配置

- Django项目中会有一些CSS、JS等静态文件，这个目录通常位于根目录下，命名为static，需要在项目的`settings.py`里做一些配置。

```
STATIC_URL = '/static/'
STATIC_ROOT = os.path.join(BASE_DIR,'static)#指明静态文件的收集目录
```

为了安全起见，在生产环境下关闭`DEBUG`选项以及设置允许访问的域名。

```
DEBUG = False
ALLOWED_HOSTS = ['127.0.0.1','localhost','39.105.110.19']
```

`ALLOWED_HOSTS`是允许访问的域名列表，前两个是本地域名，最后一个是服务器IP地址。

项目还会依赖一些第三方python库，为了在服务器上安装，我们需要将全部依赖写入`requirements.txt`文件中激活本地的虚拟环境，进入项目的根目录，运行`pip freeze >requirements.txt`命令。

```
E:\Painting>
pip freeze > requirements.txt
```

这时项目的根目录下会生成一个requirement.txt的文本文件，里面是项目的所有依赖。

将代码上传到Github

- 这里对于大多数人来说已经很熟悉了。

```
git add .
git commit -m "blog"
git push origin master
```

设置服务器目录结构

- 我的服务器上存放代码的目录结构是这样的：

```
/home/wangyufei
    sites/
        39.105.110.19/
            env/    #python虚拟环境目录
            Painting/   #项目目录
```

运行下面的命令：

```
wangyuyfei@localhost:~$ mkdir -p ~/sites/39.105.110.19
```

创建虚拟环境，进入到`39.105.110.19`目录下，运行`virtualenv`命令创建虚拟环境：

```
wangyuyfei@localhost:~$ cd ~/sites/39.105.110.19
wangyuyfei@localhost:~/sites/39.105.110.19$ virtualenv --python=python3 env
```

检查一下虚拟环境是否创建成功，运行`ls`命令查看，看到`env`这个文件夹说明虚拟环境创建成功。

```
wangyuyfei@localhost:~/sites/39.105.110.19$ ls
env
```

接着从代码仓库把项目拉取下来，`git clone`后面的地址换成自己项目的仓库地址。

```
wangyuyfei@localhost:~/sites/39.105.110.19$ git clone
https://github.com/wangyufei1006/Painting.git
```

运行`ls`命令检查是否拉取成功。

```
wangyuyfei@localhost:~/sites/39.105.110.19$ ls
AICopyBook env
```

安装项目依赖

- 激活虚拟环境，再进入到项目根目录，安装项目的全部依赖：

```
wangyuyfei@localhost:~/sites/39.105.110.19$ source env/bin/activate
(env) wangyuyfei@localhost:~/sites/39.105.110.19$ cd Painting/
(env) wangyuyfei@localhost:~/sites/39.105.110.19/Painting$ pip install -r
requirements.txt
```

收集静态文件

- 虚拟环境下继续运行`python manage.py collectstatic`命令收集静态文件到static目录下：

```
(env) wangyuyfei@localhost:~/sites/39.105.110.19/Painting$ python manage.py
collectstatic
```

生成数据库

- 虚拟环境下运行`python manage.py migrate`命令创建数据库文件：

```
(env) wangyuyfei@localhost:~/sites/39.105.110.19/Painting$ python manage.py migrate
```

创建超级用户

- 虚拟环境下运行`python manage.py createsuperuser`命令创建一个超级用户，方便进入Django管理 后台。

```
(env) wangyuyfei@localhost:~/sites/39.105.110.19/Painting$ python manage.py
createsuperuser
```

配置Nginx

- 先在服务器的`/etc/nginx/sites-available/`目录下新建一个配置文件`sudo vim 39.105.110.19`，文件名设置为域名，写上下面的配置内容：

```
/etc/nginx/sites-available/39.105.110.19

server {
    charset utf-8;
    listen 80;
    server_name 39.105.110.19;  #服务器域名

    location /static {
        alias /home/wangyufei/sites/39.105.110.19/Painting/static       #指明静态文件
存放目录
    }

    location / {
        proxy_set_header Host $host;
        proxy_pass http://unix:/tmp/39.105.110.19.socket;   #使用Unix套接字，防止端口
冲突
    }
}
```

接下来需要创建一个符号链接，把这个配置文件加入到启用的网站列表中去，被启用网站的目录在`/etc/nginx/sites-enabled/`，具体命令如下：

```
(env) wangyuyfei@localhost:~/sites/39.105.110.19/Painting$ sudo ln -s
/etc/nginx/sites-available/39.105.110.19 /etc/nginx/sites-enabled/39.105.110.19
```

使用Gunicorn

- Gunicorn一般用来管理多个进程。
- 在虚拟环境下，安装Gunicorn：

```
(env) wangyuyfei@localhost:~/sites/39.105.110.19/Painting$ pip install gunicorn
```

- 用Gunicorn启动服务器进程：

```
(env) wangyuyfei@localhost:~/sites/39.105.110.19/Painting$ gunicorn --bind
unix:/tmp/39.105.110.19.socket Painting.wsgi:application
```

浏览器输入域名，可以看到访问成功了。

- `注意`：此时如果访问到的还是Nginx的欢迎界面，需要删除`/etc/nginx/sites_available/default`和`/etc/nginx/sites_enabled/default`。

自启动Gunicorn

- 现在Gunicorn是我们手动启动的，万一哪天服务器重启了我们又得手工启动。为此写一个脚本，当服务器重新启动后，脚本会帮我们重启Gunicorn，按Ctrl+c停止服务器进程。

  写一个启动脚本，脚本位于`/etc/init`目录下，且脚本文件名必须以`.conf`结尾：

```
/etc/init/gunicorn-39.105.110.19.conf

start on net-device-up  #在服务器联网时才启动gunicorn
stop on shutdown

respawn     #重启gunicorn

setuid wangyufei    #用户名
chdir /home/wangyufei/static/39.105.110.19/Painting #进入到指定目录

exec ../env/bin/gunicorn --bind unix:/tmp/39.105.110.19.socket
Painting.wsgi:application    #执行进程
```

通过`start`命令启动gunicorn：

```
sudo start gunicorn-39.105.110.19
```

以后代码更新了，只要运行下面的命令重启一下Nginx和Gunicorn可以使新的代码生效了：

```
sudo service nginx reload
sudo restrt gunicorn-39.105.110.19
```

- 还有另外一种方法也可以实现自启动：

```
# 新建目录
sudo mkdir -p /usr/lib/systemd/system

# 新建自启动的服务文件
sudo vim /usr/lib/systemd/system/39.105.110.19.service
```

```
[Unit]
After=syslog.target network.target remote-fs.target nss-lookup.target
[Service]
# 你的用户
User=wangyufei
# 你的目录
WorkingDirectory=/home/wangyufei/sites/39.105.110.19/Painting
# gunicorn启动命令
ExecStart=/home/wangyufei/sites/env/bin/gunicorn --bind
unix:/tmp/39.105.110.19.socket Painting.wsgi:application
Restart=on-failure
[Install]
WantedBy=multi-user.target
```

```
# 启动服务
sudo systemctl start 39.105.110.19

# 添加服务到开机自动运行
sudo systemctl enable 39.105.110.19.service

# 查看进程,看看gunicorn是否已经启动,有两个进程
ps -ef | grep gunicorn
```