

# Django 搭建单服务实现多域名访问

PayneLi Python中文社区 1周前



作者：PayneLi，Python全家桶公众号小编，主要推送数据挖掘、机器学习和深度学习领域的前沿技术，同时还会推荐一些行业最新论文、技术专家的经验分享。

最近使用Django开发一个小程序和后台管理系统，需要将这两个不同的项目部署到同一个服务里面，然后使用不同的域名来访问不同的项目。Django默认的只支持单服务访问，要想实现不同域名，需要安装django的第三方扩展包：django-hosts。

本文通过一个简单的demo来给演示，看本文的前提是需要对django有一定的认识，对项目环境的搭建以及Nginx有一定的了解。

## 一.搭建环境与项目：

平时习惯使用Anconda来管理Python包，所以本文也使用anconda管理环境，当然大家要是习惯使用virtualenv管理环境也没有问题。本文主要是演示怎么完整搭建一个单服务实现多域名访问的流程，不涉及具体的业务流程。

由于生成django项目需要先下载django包，所以我们先创建一个基本的虚拟环境，然后在虚拟环境中通过具体命令来生成项目文件。

### 1、环境搭建：applitweb

```
conda create --name applite_web
```

创建完虚拟环境applite\_web了，现在需要下载一些依赖包。这里只需要单独安装django与django-hosts即可。

```
pip install django
pip install django-hosts
pip install uwsgi
```

## 2、创建项目：applite\_web

```
django-admin.py startproject applite_web
```

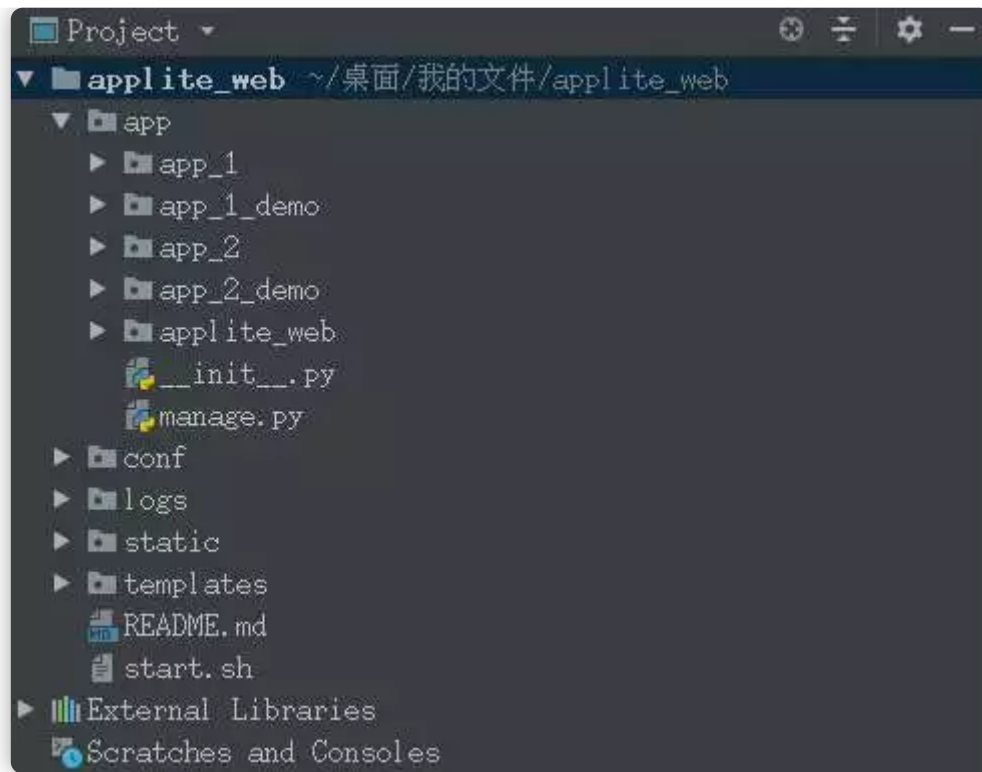
## 3、创建app

```
# 这里创建4个app.
# app_1、app_2：是用来匹配applite_web文件下,hosts.py文件分发的url
# 相当django原有的一级url.
# app_1_demo、app_2_demo：是用来定义自己的API
创建app_1:  django manage.py startapp app_1
创建app_2:  django manage.py startapp app_2
创建app_1_demo:  django manage.py startapp app_1_demo
创建app_2_demo:  django manage.py startapp app_2_demo
```

### 说明2点：

- 1)、根据项目的需要，将原来Django项目结构做了调整
- 2)、至于创建4个app的用法后边会具体涉及到

下面截图是创建本文Demo项目结构，当然这也是比较简单的一个项目结构，实际项目开发中，还需要配置别的参数与文件。



## 2. 在applite\_web中配置django-hosts

创建好虚拟环境和项目之后，接下来重点就是在django中配置多域名。为了方便起见，本文只演示2个域名，多个域名按照2个域名的方式增加就行。

### 1、首先在settins.py同级增加一个hosts.py文件，配置如下：

```
from django_hosts import patterns, host

host_patterns = patterns(
    host(r'app1', 'app_1.urls', name='app1'),
    host(r'app2', 'app_2.urls', name='app2'),
)
```

### 2、然后在django的settings配置文件增加3处配置：

- 在INSTALLED\_APPS中增加下面几个

```
INSTALLED_APPS = [
    "django_hosts",
    'app_1',
    'app_2',
    "app_1_demo",
    "app_2_demo",
]
```

- 需要在 MIDDLEWARE 的首行和末行增加2行配置

```
MIDDLEWARE = [
    'django_hosts.middleware.HostsRequestMiddleware',
    .....
    'django_hosts.middleware.HostsRequestMiddleware',
]
```

- 在ROOT\_URLCONF下行增加如下2行参数配置

```
# 客户端的请求通过这个配置被转发到hoss.py文件的host_patterns中匹配
ROOT_HOSTCONF = 'applite_web.hosts'
# 设置一个默认域名, 在没有匹配所有请求的域名时, 默认请求这个域名
DEFAULT_HOST = 'app1'
```

### 3. 配置url和view

上边两步操作完成, 接下来就需要去app文件夹下, 编写对应的url和view。本文是演示2个域名, 同时考虑到一般单个项目会有多个模块, 所以还需要分别配置两个url。具体的配置如下:

#### 1、分别在app\_1、app\_2文件夹下创建urls.py文件, 并且增加对应的url,代码如下:

- app\_1的urls.py文件如下:

```
from django.urls import path, include

urlpatterns = [
    path("app1/", include("app_1_demo.urls"))
]
```

- app\_2的urls.py文件如下

```
from django.urls import path, include

urlpatterns = [
    path("app2/", include("app_2_demo.urls")),
]
```

#### 2、然后在app\_1\_demo、app\_2\_demo中分别增加对应的url和views函数

- 在app\_1中匹配成功之后, 会直接跳转到app\_1\_demo的urls, 再根据当前的匹配跳转当前的views函数

##### 1)、首先匹配app\_1\_demo的urls的路由:

```
from django.urls import path
from .views import app_1_view

urlpatterns = [
```

```
path("", app_1_view),  
]
```

## 2)、urls匹配成功之后, 跳转这里的视图函数, 并返回响应

```
from django.http.response import HttpResponseRedirect  
  
# Create your views here.  
def app_1_view(request):  
    return HttpResponseRedirect("hello i'm app_1")
```

- 同理, app\_2匹配成功之后, 会直接跳转到app\_2\_demo的urls, 再根据当前的匹配跳转当前的views函数

## 1)、首先匹配app\_2\_demo的urls的路由, 匹配如下:

```
from django.urls import path, include  
from .views import app_2_view  
  
urlpatterns = [  
    path("", app_2_view)  
]
```

## 2)、urls匹配成功之后, 跳转这里的视图函数, 并返回响应

```
from django.http.response import HttpResponseRedirect  
  
# Create your views here.  
def app_2_view(request):  
    return HttpResponseRedirect("hello i'm app_2")
```

## 4.域名绑定与测试

经过上边三步的操作, 已经完成了一个基本的演示功能。本文目的是通过Nginx负载单服务后, 在一个服务里面来实现不同域名的访问, 所以在配置uwsgi和Nginx参数, 还需要绑定域名。

### 1、这里是在同一个局域网下通过两台机器实现用户的访问:

访问机器IP: 192.168.2.17, 部署服务的IP: 192.168.2.200。在实际生产需要购买真实的域名, 这里作为演示, 可以通过在本机绑定服务器的ip的方式来实现对另一台机器服务的访问, 具体修改如下:

```
# 需要进入到此文件中编辑增加下面两行: sudo vim /etc/hosts
# 这里我们分别给192.168.2.200绑定app1.cc与app2.cc两个域名

192.168.2.200 app1.cc
192.168.2.200 app2.cc
```

2、绑定完之后，然后将本项目放到192.168.2.200机器上，测试当前的服务配置没有问题。  
项目放置路径为： /home/yxy/payneli/applite\_web/

- 进入app文件夹下，然后运行项目，命令如下：

```
python manage.py runserver 192.168.2.200:8000
```

- 浏览器输入： app1.cc:8000/app1/，如果浏览器显示如下结果，说明项目配置与域名绑定成功



## 5.uwsgi参数配置与测试

前边几步成功之后，接下来就是配置uwsgi的参数。熟悉Python后台开发的都应该清楚，开发的时候使用的 `python manage.py runserver` 来运行服务器，这只适用开发时的代码调试，而实际项目部署的话，django内置的服务根本无法满足需求。而Uwsgi作为Python服务器不仅可以提供稳定的服务，同时还可以提供大的并发量，所以在后台开发中，用的比较多。

1、在项目文件夹下，创建一个uwsgi文件夹。进入该文件夹，创建uwsgi.ini文件，里面的配置参数如下：

```
[uwsgi]
# 项目目录
chdir=/home/yxy/payneli/applite_web/app/
# 指定项目的application
wsgi-file=applite_web/wsgi.py
# 指定sock的文件路径
```

```

socket=/home/yxy/payneli/applite_web/uwsgi/uwsgi.sock
# 进程个数
workers=1
pidfile=/home/yxy/payneli/applite_web/uwsgi/uwsgi.pid
# 指定IP端口
# nginx负载均衡使用socket,uwsgi启动服务使用http
#socket=192.168.2.200:8000
http=192.168.2.200:8000
# 启用主进程
master=true
# 自动移除unix Socket和pid文件当服务停止的时候
vacuum=true
# 序列化接受的内容, 如果可能的话
thunder-lock=true
# 启用线程
enable-threads=true
# 设置自中断时间
harakiri=30
# 设置缓冲
post-buffering=4096
# 设置日志目录
daemonize=/home/yxy/payneli/applite_web/logs/uwsgi.log

```

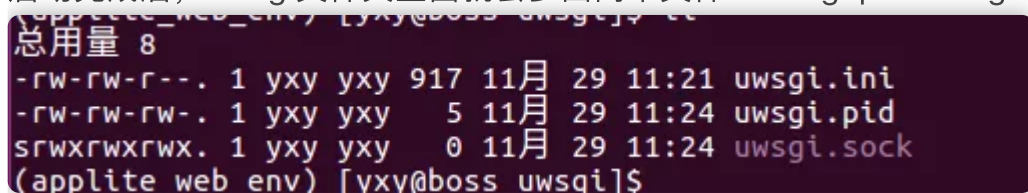
2、配置完成后, 运行下面的命令, 启动服务。

```

# 在项目路径下, 启动服务命令如下:
uwsgi --ini ./uwsgi/uwsgi.ini

```

启动完成后, uwsgi文件夹里面就会多出两个文件: uwsgi.pid uwsgi.sock



```

总用量 8
-rw-rw-r--. 1 yxy yxy 917 11月 29 11:21 uwsgi.ini
-rw-rw-rw-. 1 yxy yxy  5 11月 29 11:24 uwsgi.pid
srwxrwxrwx. 1 yxy yxy  0 11月 29 11:24 uwsgi.sock
(applite web env) [yxy@boss uwsgi]$

```

然后查看当前服务是否启动成功:

```

# 命令如下
ps -ef|grep uwsgi

```

当显示如下图所示的时候, 说明uwsgi已经将项目启动成功



```
[uwsgi] getting INI configuration from uwsgi.ini
(applite_web_env) [yxy@boss uwsgi]$ ps -ef|grep uwsgi
yxy      3845      1    1 11:24 ?        00:00:00 uwsgi --ini uwsgi.ini
yxy      3847    3845    0 11:24 ?        00:00:00 uwsgi --ini uwsgi.ini
yxy      3848    3845    0 11:24 ?        00:00:00 uwsgi --ini uwsgi.ini
yxy      3850    2878    0 11:24 pts/1    00:00:00 grep --color=auto uwsgi
```

### 3、浏览器测试服务是否正常：

在浏览为直接输入：app1.cc:8000/app1/,如果显示如下，说明uwsgi配置成功



## 6. Nginx配置与测试

如果前边5步没有问题的话，那么恭喜你，就差最后一步就可以完成本demo的演示了。现在就开始最后一步，配置Nginx参数。

1、首先安装Nginx，本文对nginx安装不做讲解，毕竟网上教程那么多，可以找一个好的教程照着操作就可以。小编将Nginx安装为默认路径，在：/usr/local/nginx，进入此文件下，直接命令行启动。

```
/usr/local/nginx/sbin/nginx
```

查看Nginx是否启动成功，命令如下：

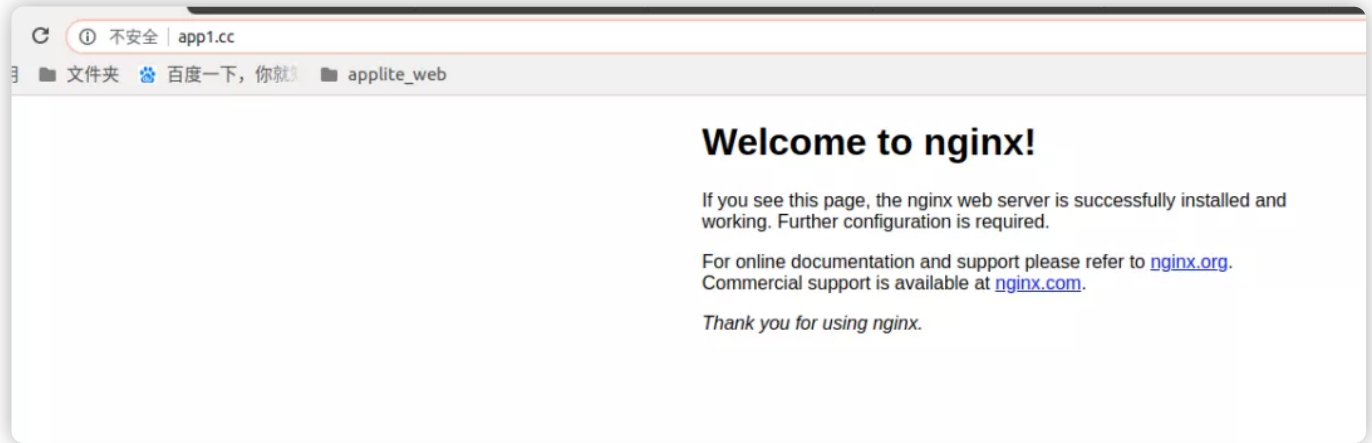
```
ps -ef|grep nginx
```

显示如下图，说明Nginx启动成功

```
(applite_web_env) [yxy@boss nginx]$ ps -ef|grep nginx
root      4101      1    0 15:07 ?        00:00:00 nginx: master process ./sbin/nginx
root      4102    4101    0 15:07 ?        00:00:00 nginx: worker process
root      4103    4101    0 15:07 ?        00:00:00 nginx: worker process
root      4104    4101    0 15:07 ?        00:00:00 nginx: worker process
root      4105    4101    0 15:07 ?        00:00:00 nginx: worker process
yxy       4107    2878    0 15:08 pts/1    00:00:00 grep --color=auto nginx
(applite_web_env) [yxy@boss nginx]$
```



浏览器输入：app1.cc，显示如下所示，说明Nginx安装成功



2、检查Nginx配置没有问题之后，就是配置多域名了。

- 注释掉Nginx原有的配置server

```
#gzip on;
#server {
#    listen 80;
#    server_name localhost;
#
#    location / {
#        root html;
#        index index.html index.html;
#    }
# }
```

- 增加如下一行参数，主要是为了方便单独增加配置app1.cc、与app2.cc两个域名的文件

```
#gzip on;
#server {
#    listen 80;
#    server_name localhost;
#
#    location / {
#        root html;
#        index index.html index.html;
#    }
# }

include /usr/local/nginx/conf/multihosts/*.conf;
```

- 分别在 /usr/nginx/conf/文件加下，创建一个新的multihosts文件夹，进入该文件夹分别新建 app1.cc.conf、app2.cc.conf文件，具体的配置参数如下：

**app1.cc.conf配置如下：**

```

server{
    listen 80;    # 默认监听80端口
    server_name app1.cc; # 访问的域名
    #index index.html inex.htm index.php;
    #root /data/www/applite_web/;
    access_log /usr/local/nginx/logs/app1.log logmain;

    rewrite_log on;
    #error_page 404 /404.html;

    location / {    # 通用匹配, 任何未匹配到其它location的请求都会匹配到
        include uwsgi_params; # uwsgi参数
        uwsgi_pass 192.168.2.200:8000; # 负载后台服务
        uwsgi_param UWSGI_CHDIR /home/yxy/payneli/applite_web/app/;
        uwsgi_param UWSGI_SCRIPT applite_web.wsgi;
        client_max_body_size 35m;
        uwsgi_send_timeout 1060;
        uwsgi_connect_timeout 1060;
        uwsgi_read_timeout 1060;
    }
}

```

- app2.cc.conf配置如下:

```

server{
    listen 80;
    server_name app2.cc;
    #index index.html inex.htm index.php;
    #root /home/yxy/www/applite_web/;
    access_log /usr/local/nginx/logs/app2.log logmain;

    rewrite_log on;
    #error_page 404 /404.html;

    location / {
        include uwsgi_params;
        uwsgi_pass 192.168.2.200:8000;
        uwsgi_param UWSGI_CHDIR /home/yxy/payneli/applite_web/app/;
        uwsgi_param UWSGI_SCRIPT applite_web.wsgi;
        client_max_body_size 35m;
        uwsgi_send_timeout 1060;
        uwsgi_connect_timeout 1060;
        uwsgi_read_timeout 1060;
    }
}

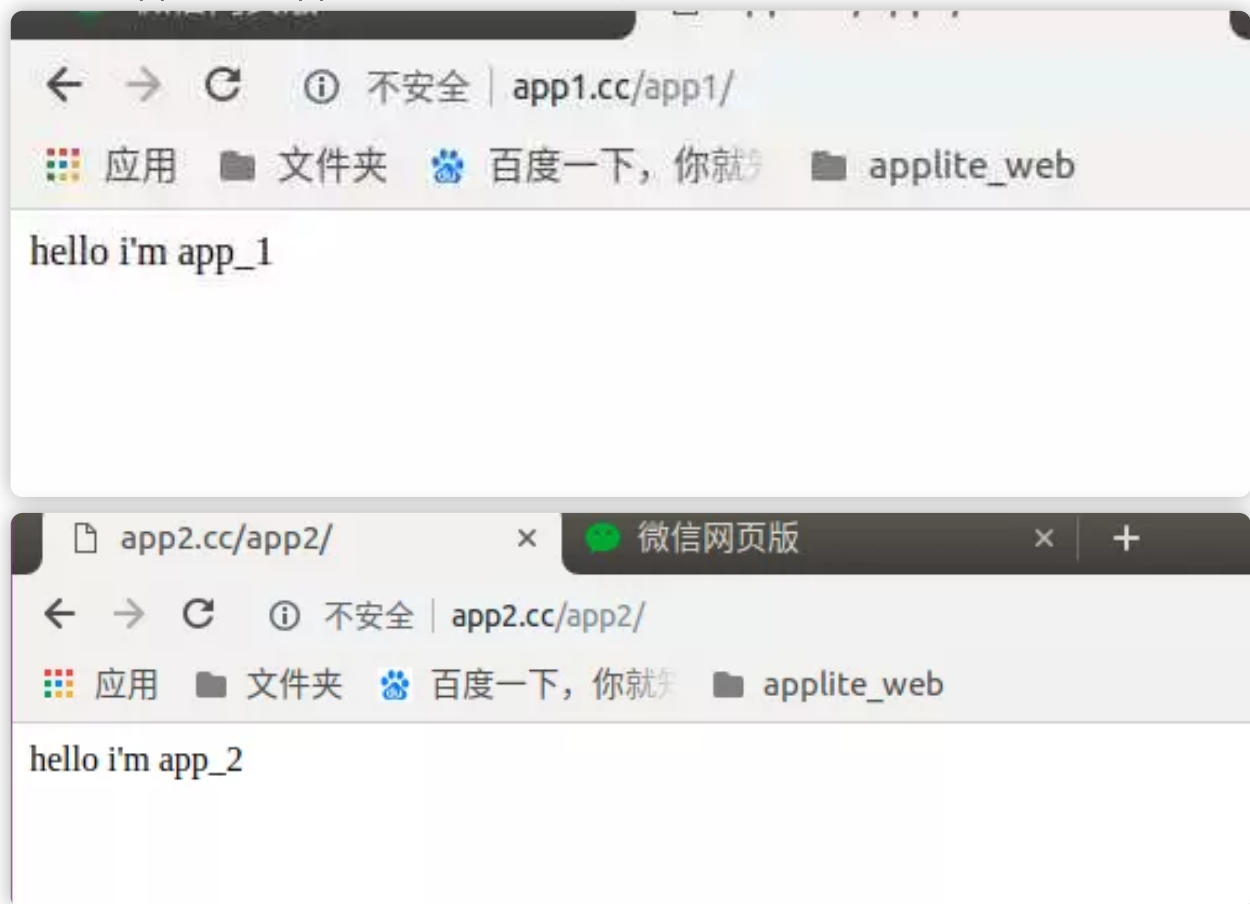
```

3、多域名的参数配置完成，但是此刻不要忘了，当使用Nginx作为负载均衡时候，需要将uwsgi.ini里面的参数http改为socket具体如下：

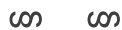
```
# nginx负载均衡使用socket,uwsgi启动服务使用http
socket=192.168.2.200:8000
# http=192.168.2.200:8000
```

4、到这里本文的参数已经配置完成，然后我们需要重新启动Nginx，查看显示如上边启动效果，说明启动成功，接下来就是浏览器检测多域名配置是否正确：

分别输入：app1.cc、app2.cc显示结果如下，说明多域名已经配置成功



到此为止，单服务实现多域名访问已经演示完成。当然，本文只是做了一个简单的demo版本，而实际项目开发的过程中，还需要根据实际需要评估，决定到底是nginx后台负载多服务，还是nginx负载单服务映射多域名。



Python中文社区作为一个去中心化的全球技术社区，以成为全球20万Python中文开发者的精神部落为愿景，目前覆盖各大主流媒体和协作平台，与阿里、腾讯、百度、微软、亚马逊、开源中国、CSDN等业界知名公司和技术社区建立了广泛的联系，拥有来自十多个国家和地区数万名登记会员，会员来自以公安部、工信部、清华大学、北京大学、北京邮电大学、中国人民银行、中科院、中金、华为、BAT、谷歌、微软等为代表的政府机关、科研单位、金融机构以及海内外知名公司，全平台近20万开发者关注。

## 最近热门

搭建CNN模型破解网站验证码

用Python进行图文识别（OCR）

Python开发在北京的就业现状分析

用Python回忆QQ空间里的青春

投稿邮箱：[pythonpost@163.com](mailto:pythonpost@163.com)



# Python中文社区

全球20万Python中文开发者的精神部落

合作微信: AndyWong188

投稿邮箱: pythonpost@163.com



长按扫码关注

GitHub: [github.com/PyCN](https://github.com/PyCN)

▼ 请点击下方阅读原文, 查看[Python中文社区全部文章](#)

阅读原文