五步教你实现使用Nginx+uWSGI+Django方法部署Django程序(上)

By [Django中国社区](http://django-china.cn/user/1/) at 2013-05-12 15:05

Django的部署可以有很多方式，采用nginx+uwsgi的方式是其中比较常见的一种方式。

[](http://www.th7.cn/Article/UploadFiles/200808/2008081305323204.jpg)

在这种方式中，我们的通常做法是，将nginx作为服务器最前端，它将接收WEB的所有请求，统一管理请求。nginx把所有静态请求自己来处理（这是NGINX的强项）。然后，NGINX将所有非静态请求通过uwsgi传递给Django，由Django来进行处理，从而完成一次WEB请求。

可见，uwsgi的作用就类似一个桥接器。起到桥梁的作用。

NOTE：不使用nginx，只使用uwsgi+django也是可以实现WEB服务的。uwsgi也可以直接处理WEB请求。

为了完成上述的方式部署，我将分成两篇文章来分别进行阐述。

* 第一步先解决uwsgi与django的桥接。解决在没有nginx的情况下，如何使用uwsgi+DJANGO来实现一个简单的WEB服务器。
* 第二步解决uwsgi与Nginx的桥接。通过nginx与uwsgi的桥接，打通nginx与django的连通，从而比较完美的实现django的部署。

本文将分成五步来详细阐述uwsgi+django的部署方式。nginx+uwsgi+django的部署将在下一篇 文章中阐述。

本文大纲：

* 环境介绍
* 安装uwsgi
* 测试uwsgi
* 配置django
* 连接django和uwsgi，实现简单的WEB服务器。

**环境介绍**

1. Ubuntu 12.04.1 LTS
2. django 1.4.2

**安装uwsgi**

1.安装pip

可以参考这篇文章：<http://www.jsxubar.info/install-pip.html>

2.安装uwsgi

$ export LDFLAGS="-Xlinker --no-as-needed"

$ pip install uwsgi

如果安装出现错误，则先执行

apt-get install build-essential python-dev

**测试uwsgi**

在你的机器上写一个test.py

# test.py

def application(env, start\_response):

start\_response('200 OK', [('Content-Type','text/html')])

**return** "Hello World"

然后执行shell命令：

uwsgi --http :8001 --wsgi-file test.py

访问网页：

<http://127.0.0.1:8001/>

看在网页上是否有Hello World

**配置django**

NOTE：

请保证你的django项目是正常使用的。可以使用

python manage.py runserver 0.0.0.0:8002

来测试一下你的django项目是否能正常跑起来。

请保证你的django程序已经关闭。

编写django\_wsgi.py文件，将其放在与文件manage.py同一个目录下。

注意： 编写文件时需要注意语句os.environ.setdefault。比如，如果你的项目为mysite，则你的语句应该是 os.environ.setdefault("DJANGO\_SETTINGS\_MODULE", "mysite.settings")

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | *#!/usr/bin/env python*  *# coding: utf-8*  **import** **os**  **import** **sys**  *# 将系统的编码设置为UTF8*  reload(sys)  sys.setdefaultencoding('utf8')  os.environ.setdefault("DJANGO\_SETTINGS\_MODULE", "mysite.settings")  **from** **django.core.handlers.wsgi** **import** WSGIHandler  application = WSGIHandler() |

**连接django和uwsgi，实现简单的WEB服务器。**

我们假设你的Django项目的地址是/home/work/src/sites/testdjango1/testdjango/mysite，

然后，就可以执行以下命令：

uwsgi --http :8000 --chdir /home/work/src/sites/testdjango1/testdjango/mysite --module django\_wsgi.py

这样，你就可以在浏览器中访问你的Django程序了。所有的请求都是经过uwsgi传递给Django程序的。

**最后：**

关于如何将uwsgi与Nginx连接，可以期待下篇文章 《五步教你实现使用Nginx+Uwsgi+Django方法部署Django程序(下)》

最后面，请大家要支持[Django中国社区](http://django-china.cn/)哦，单靠一两个人是不行的，一起推广一下，让Django社区更有力量哈！更有人气哈！

推广链接： <http://django-china.cn/>

五步教你实现使用Nginx+uWSGI+Django方法部署Django程序(下)

By [Django中国社区](http://django-china.cn/user/1/) at 2013-05-18 04:38

在上一篇文章[《五步教你实现使用Nginx+uWSGI+Django方法部署Django程序(上)》](http://django-china.cn/topic/101/)中，阐述了如何只使用uWSGI来部署Django程序。

当然，单单只有uWSGI是不够的，在实际的部署环境中，Nginx是必不可少的工具。

在本篇文章中，我将一直延用“N步法”的风格来阐述如何将uWSGI与Nginx做连接来部署Django程序。并在最后，会较为完整的阐述本社区的部署方法。

**本文大纲：**

* 环境介绍
* 配置uWSGI
* 配置Nginx
* Nginx+uWSGI+Django的实现方式
* 一些建议

**环境介绍**

1. Ubuntu 12.04.1 LTS
2. django 1.4.2
3. nginx/1.2.6
4. uWSGI 1.4.4

关于uWSGI的安装可参见上一篇文章 上一篇文章[《五步教你实现使用Nginx+uWSGI+Django方法部署Django程序(上)》](http://django-china.cn/topic/101/)

我们假定你已经安装好Nginx了。

**配置uWSGI**

在上一篇文章[《五步教你实现使用Nginx+uWSGI+Django方法部署Django程序(上)》](http://django-china.cn/topic/101/)中,我们是直接使用命令行来启动uWSGI，在实际部署环境中，我们常用的是配置文件的方式，而非命令行的方式。我的一般做法是用命令行来测试是否uWSGI安装成功，然后用配置文件来真正部署。

另外，为了实现Nginx与uWSGI的连接，两者之间将采用soket来通讯方式。

在本节中，我们将使用uWSGI配置文件的方式来改进uWSGI的启动方式。

假定你的程序目录是 /home/work/src/sites/testdjango1/testdjango/mysite

我们将要让Nginx采用8077端口与uWSGI通讯，请确保此端口没有被其它程序采用。

注意，请确定你在上一节[《五步教你实现使用Nginx+uWSGI+Django方法部署Django程序(上)》](http://django-china.cn/topic/101/)中的django\_wsgi.py文件已经存在了。

新建一个XML文件：

djangochina\_socket.xml，将它放在 /home/work/src/sites/testdjango1/testdjango/mysite 目录下：

**<uwsgi>**

**<socket>**:8077**</socket>**

**<chdir>**/home/work/src/sites/testdjango1/testdjango/mysite**</chdir>**

**<module>**django\_wsgi**</module>**

**<processes>**4**</processes>** *<!-- 进程数 -->*

**<daemonize>**uwsgi.log**</daemonize>**

**</uwsgi>**

在上面的配置中，我们使用 uwsgi.log 来记录日志，开启4个进程来处理请求。

这样，我们就配置好uWSGI了。

**配置Nginx**

我们假设你将会把Nginx程序日志放到你的目录/home/work/var/test/logs/下，请确保该目录存在。

我们假设你的Django的static目录是/home/work/src/sites/testdjango1/testdjango/collectedstatic/ ， media目录是/home/work/src/sites/testdjango1/testdjango/public/media/，请确保这些目录存在。

我们假设你的域名是 [www.you.com](http://www.you.com/) （在调试时你可以设置成你的机器IP）

我们假设你的域名端口是 80（在调试时你可以设置一些特殊端口如 8070）

基于上面的假设，我们为conf/nginx.conf添加以下配置

server {

listen 80;

server\_name www.you.com;

access\_log /home/work/var/test/logs/access.log;

error\_log /home/work/var/test/logs/error.log;

#charset koi8-r;

#access\_log logs/host.access.log main;

location / {

include uwsgi\_params;

uwsgi\_pass 127.0.0.1:8077;

}

#error\_page 404 /404.html;

# redirect server error pages to the **static** page /50x.html

#

error\_page 500 502 503 504 /50x.html;

location = /50x.html {

root html;

}

location /**static**/ {

alias /home/work/src/sites/testdjango1/testdjango/collectedstatic/;

index index.html index.htm;

}

location /media/ {

alias /home/work/src/sites/testdjango1/testdjango/public/media/;

}

}

在上面的设置后，可以让Nginx来处理静态文件(/static/ 和 /media/ ）。非静态文件请求Nginx会发给 socket 8077，然后让uWSGI来进行处理。

**Nginx+uWSGI+Django的实现方式**

在完成上面配置后，需要按以下步骤来做：

1. 重启Nginx服务器，以使Nginx的配置生效。
2. nginx -s reload

重启后检查Nginx日志是否有异常。

1. 启动uWSGI服务器
2. cd /home/work/src/sites/testdjango1/testdjango/mysite
3. uwsgi -x djangochina\_socket.xml

检查日志 uwsgi.log 是否有异常发现。

1. 访问服务

基于上面的假设你的域名是[www.you.com](http://www.you.com/)

因此，我们访问 [www.you.com](http://www.you.com/)，如果发现程序与 单独使用Django启动的程序一模一样时，就说明成功啦！

1. 关闭服务的方法

将uWSGi进程杀死即可。

**一些建议**

1. uWSG配置文件的进程数，可以根据实际情况分配。不要开得太大，否则机器可能会内存耗用太高。一般来说，对于一个小社区来说，4个进程已经足够了。
2. 一般情况下，可以编写一下 stop.sh 脚本 来关闭uWSGI。

**最后**

最后面，请大家要支持[Django中国社区](http://django-china.cn/)哦，单靠一两个人是不行的，一起推广一下，让Django社区更有力量哈！更有人气哈！

推广链接： <http://django-china.cn/>

**END**