# Linux下安装Apache并以mod\_wsgi方式部署django站点

### 源码编译方式安装Apache

首先下载Apache源码压缩包，地址为http://mirror.bit.edu.cn/apache/httpd/

继续下载apr和apr-util压缩包，地址为http://mirror.bit.edu.cn/apache/apr/

下载pcre压缩包，地址为http://ftp.exim.llorien.org/pcre/

l 安装arp

tar -zxf apr-1.5.0.tar.gz #解压压缩包 压缩包名称根据版本号决定cd apr-1.5.0 #进入解压目录./configure --prefix=/usr/local/apr #进行安装配置 设置安装位置make #编译sudo make install #安装

l 安装apr-util

tar –zxvf apr-util-1.5.3.tar.gz #解压 压缩包名称根据版本号决定cd apr-util-1.5.3 #进入解压目录./configure --prefix=/usr/local/apr-util --with-apr=/usr/local/apr/bin/apr-1-configmake sudo make install

l 安装pcre

unzip –o pcre-8.34.zip #解压 文件名依据版本号而定cd pcre-8.34./configure --prefix=/usr/local/pcre

make sudo make install

----------------------------------------------------------------------------------

,执行make 操作时出现 如下问题

   Makefile:372: recipe for target 'aclocal.m4' failed

解决办法：在linux终端下执行 ：autoreconf -vfi

问题：autoreconf: failed to run aclocal: 没有那个文件或目录

解决： sudo yum install automake

-------------------------------------------------------------------------------

l 安装Apache

tar –zxvf httpd-2.4.7.tar.gz #解压 我安装的为2.4.7版本cd httpd-2.4.7

./configure --prefix=/usr/local/apache2 --with-apr=/usr/local/apr --with-apr-util=/usr/local/apr-util/ --with-pcre=/usr/local/pcre/

make sudo make install

sudo httpd start

### 如何使Apache支持django

采用mod\_wsgi的方式。

首先需要下载mod\_wsgi-2.4.tar.gz。

可以到官网下载。

还可以到点这里直接下载。

接下来执行如下命令，对压缩包进行解压；

tar –zxvf mod\_wsgi-3.4.tar.gz

然后cd到解压之后的目录，执行如下命令，产生配置编译配置文件；

./configure --with-apxs=/usr/local/apache2/bin/apxs --with-python=/usr/bin/python

============================================================

## mod\_python 编译出错的解决办法

下载源码包，解压得到目录，进入该目录，执行：

 ./configure --with-apxs=/usr/local/apache2/bin/apxs --with-python=/usr/bin/python2.4

 make 后出错退出

connobject.c: In function '\_conn\_read':

connobject.c:142: error: request for member 'next' in something not a structure

or union

apxs:Error: Command failed with rc=65536

.

make[1]: \*\*\* [mod\_python.so] Error 1

make[1]: Leaving directory `/usr/src/rpm/BUILD/mod\_python-3.3.1/src'

make: \*\*\* [do\_dso] Error 2

这是一个bug，作如下修改

需修改mod\_python-3.3.1/src/connobject.c中

!(b == APR\_BRIGADE\_SENTINEL(b)

为

!(b == APR\_BRIGADE\_SENTINEL(bb)

完毕之后，mod\_python 会在 /usr/local/apache2/modules 中添加 mod\_python.so 库文件，并且在 /usr/lib/python2.4/site-packages/中添加 mod\_python 包。

如果编译程序没有检测到 apache 和 python，会报错退出。

===========================================================

其中/usr/local/apache2为Apache的安装目录，确保apxs存在于bin目录中，/usr/bin/python为Ubuntu中python默认安装位置。

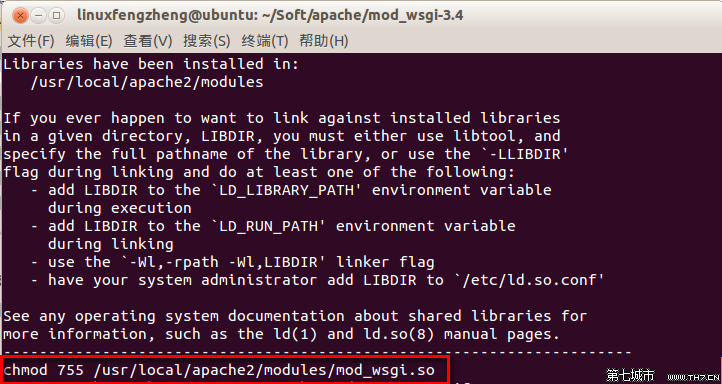
接下来执行make 命令，进行编译；

make

最后执行如下命令，进行安装；

sudo make install

如果到这里都顺利的话，会提示“chmod 755 /usr/local/apache2/modules/mod\_wsgi.so” 为mod\_wsgi.so赋权。如图：



照着提示输入命令即可：

chmod 755 /usr/local/apache2/modules/mod\_wsgi.so

这几天花了点时间,将把django开发好的web项目部署到Apache上，参考了官方的一些文档和互联网上的文档，还是花了比较多的时间，这里把配置的过程说一下。  
方便有需要的朋友，可以参考，少走弯路!  
**1. django项目部署环境说明**操作系统 : Red Hat Enterprise Linux Server release 5.3 (Tikanga) x86\_64  
apache版本 : httpd-2.2.3-22.el5  
mod\_wsgi版本 : mod\_wsgi-3.2-1.el5 fedora epel可以下载  
Django版本 : 1.2.3  
python 版本 : 2.5  
这里假定Django和Apache已经安装好，并且Django的项目已经开发好。  
以上软件包都是通过yum包安装,软件包都是系统标准目录结构!   
django开发好的项目目录是 /var/www/html/server,项目目录结构如下(标准django项目目录结构)

#tree -d server/

server/

|-- \_\_init\_\_.py

|-- manage.py

|-- settings.py

|-- backend

|-- static

| |-- images

| |-- locale

| |-- plugins

| `-- themes

| |-- default

| | `-- images

| |-- gray

| | `-- images

| `-- icons

|-- template

`-- view

**2. Apache和mod\_wsgi配置**修改wsgi配置(/etc/httpd/conf.d/wsgi.conf)

#cat /etc/httpd/conf.d/wsgi.conf

LoadModule wsgi\_module modules/mod\_wsgi.so

WSGIScriptAlias / "/var/www/html/server/django.wsgi"

<Directory "/var/www/html/server">

Order Deny,Allow

Allow from all

</Directory>

项目目录中的django.wsgi这个文件是需要新建的，后面会说到如何新建这个文件。  
apache使用的标准配置,Apache的DocumentRoot 指向的是 /var/www/html 目录  
**3. 新建django.wsgi文件**在项目目录/var/www/html/server下新建一个django.wsgi，文件内容如下:

#cat /var/www/html/server/django.wsgi

# -\*- coding: utf-8 -\*-

import os

import sys

os.environ['DJANGO\_SETTINGS\_MODULE'] = 'settings'

os.environ['PYTHON\_EGG\_CACHE'] = '/tmp/.python-eggs'

current\_dir = os.path.dirname(\_\_file\_\_)

if current\_dir not in sys.path: sys.path.append(current\_dir)

import django.core.handlers.wsgi

application = django.core.handlers.wsgi.WSGIHandler()

第三行 os.environ['DJANGO\_SETTINGS\_MODULE'] = 'settings' ，这个settings就是指项目目录下的setting.py文件.  
第四行 os.environ['PYTHON\_EGG\_CACHE'] = '/tmp/.python-eggs' ，指定解压egg文件的cache目录，确保运行apache的用户，能够对此目录有读写权限.  
第五，六行 将当前目录自动加入到python的搜索路径,如果项目中有自己写的模块，方便使用和发布　　  
最后，这个django.wsgi文件名可以随便取，例如test.wsgi、app.wsgi等等，但是一定要与/etc/httpd/conf.d/wsgi.conf配置文件中配置保持一致。  
如果您这里新建的文件名不是django.wsgi而是test.wsgi，那么/etc/httpd/conf.d/wsgi.conf中的配置就应该修改为

WSGIScriptAlias / "/var/www/html/server/test.wsgi"

**4. 修改django项目中的setting.py文件**找到项目目录下的setting.py，对于这里就是/var/www/html/server/setting.py。找到其中的TEMPLATE\_DIRS，修改为：

TEMPLATE\_DIRS = ("/var/www/html/server/template",)

注意: 模板目录在这里一定要用绝对路劲，而不能用相对路径,当然也有方法动态设置模板路劲

**PS：关于mod\_wsgi**目前mod\_wsgi有两种工作模式：

第一种是嵌入模式，类似于mod\_python，直接在apache进程中运行，这样的好处是不需要另外增加进程，但是坏处也很明显，所有内存都和apache共享，如果和mod\_python一样造成内存漏洞的话，就会危害整个apache。而且如果apache是用worker mpm，mod\_wsgi也就强制进入了线程模式，这样子对于非线程安全的程序来说就没法用了。

这种模式下需要在apache的vhost中如下设置：

<span style="font-family: 'times new roman', times;">WSGIScriptAlias /path /path-to-wsgi</span>

即可生效，对于小型脚本的话，直接用这种模式即可。

第二种是后台模式，类似于FastCGI的后台，mod\_wsgi会借apache的外壳，另外启动一个或多个进程，然后通过socket通信和apache的进程联系。

这种方式只要使用以下配置即可：

#启动WSGI后台，site1是后台名字

WSGIDaemonProcess site1 processes=1 threads=15 display-name=%{GROUP}

#分配当前上下文应该使用哪个WSGI后台，可以放在Location里面指定

WSGIProcessGroup site1

#根据当前上下文的ProcessGroup分配到对应的后台

WSGIScriptAlias /path /path-to-wsgi

在这种模式下，我们可以通过调节processes和threads的值来设置三种MPM的模式：prefork', 'worker', 'winnt'。

winnt模式

WSGIDaemonProcess example threads=25

wsgi.multithread True

wsgi.multiprocess False

此时processes=1，但是multiprocess为false

如果显式地指出processes为1那么：

WSGIDaemonProcess example processes=1 threads=25

wsgi.multithread True

wsgi.multiprocess True

worker模式

WSGIDaemonProcess example processes=2 threads=25

wsgi.multithread True

wsgi.multiprocess True

preforker模式

WSGIDaemonProcess example processes=5 threads=1

wsgi.multithread False

wsgi.multiprocess True

后台模式由于是与apache进程分离了，内存独立，而且可以独立重启，不会影响apache的进程，如果你有多个项目（django），可以选择建立多个后台或者共同使用一个后台。

比如在同一个VirtualHost里面，不同的path对应不同的django项目，可以同时使用一个Daemon：

<span style="font-family: 'times new roman', times;">WSGIDaemonProcess default processes=1 threads=1 display-name=%{GROUP}

WSGIProcessGroup default

WSGIScriptAlias /project1 “/home/website/project1.wsgi”

WSGIScriptAlias /project2 “/home/website/project2.wsgi”</span>

这样子两个django都使用同一个WSGI后台。

也可以把不同的项目分开，分开使用不同的后台，这样开销比较大，但就不会耦合在一起了。

display-name是后台进程的名字，这样方便重启对应的进程，而不需要全部杀掉。

<span style="font-family: 'times new roman', times;">WSGIDaemonProcess site1 processes=1 threads=1 display-name=%{GROUP}

WSGIDaemonProcess site2 processes=1 threads=1 display-name=%{GROUP}

<Location “/project1″>

WSGIProcessGroup site1

</Location>

WSGIScriptAlias /project1 “/home/website/project1.wsgi”

<Location “/project1″>

WSGIProcessGroup site2

</Location>

WSGIScriptAlias /project2 “/home/website/project2.wsgi”</span>

对于django 1.0以下的版本，由于官方认定不是线程安全的，所以建议使用多进程单线程模式

processes=n threads=1

对于django 1.0以后，就可以放心的使用多进程多线程模式：

processes=2 threads=64

这样子性能会更好。