

# Installing mod\_python and Django on Apache

I've recently started learning **Django** for doing web development, so as part of that, I needed to also learn how to set up Django within mod\_python so I could deploy my new Django applications on my server.

In this post, I'll be sharing the method I used to install mod\_python and Django on my CentOS 5, Apache-based web server. So without any further ado, let's get started on the process!

## Starting Off

I'll be running the latest SVN release of Django, which provides the latest features. Despite not officially being stable, I've found it to be fine, and doesn't restrict me to having to develop to the older spec of the 0.9x releases. That said, you should regularly update your SVN copy to keep updated with the latest security and bug fixes.

Apparently:

*"We improve Django almost every day and are pretty good about keeping the code stable. Thus, using the latest development code is a safe and easy way to get access to new features as they're added."*

Before we get started downloading Django, however, we first need to grab mod\_python (the plumbing between Apache and Django) for the installed Apache. The easiest way to do this is through yum:

```
# yum install mod_python
```

This should automatically add to your httpd.conf Modules, but if it doesn't, then you should manually do so with the line:

```
Module python_module modules/mod_python.so
```

## Getting and Installing Django

Now let's go and get the latest version of Django. For the whole of this tutorial, I'll assume you are putting Django in /opt, as I am here. As root:

```
# mkdir /opt/django
# cd /opt/django
# svn co http://code.djangoproject.com/svn/django/trunk/
```

Once the SVN checkout completes, you have Django downloaded. Next, we need to get it installed into Python, by symlinking this new Django directory into Python's site-packages directory.

```
# ln -s /opt/django/trunk/django /usr/lib/python2.4/site-packages
```

Also, we should at this point add **django-admin.py** to your PATH, so you can use it to create new Django projects from anywhere on your system.

```
# ln -s /opt/django/trunk/django/bin/django-admin.py
/usr/local/bin/django-admin.py
```

To test that the Django installation worked, run the following. If it works, Python shouldn't give you any error message and go silently back to its prompt. If you get **ImportError: No module named django**, you have a problem. Check that symlink to **/opt/django/trunk/django** is in Python's site-packages directory.

```
# python
>>> import django
```

May 19, 2008



Peter  
Upfold

FOSSwire

All articles

Custom Search

Search

## Getting a Database Library

You need a Python module for the database you plan to use with Django installed, so that Django can talk to the database. Most people will be using MySQL, so that's what I'll cover installing here.

Right now, the version of connector module **MySQL-python** in the CentOS repositories is too old to use with the SVN release of Django. Instead, get the latest tar.gz release from [here](#).

Once that's downloaded, do this in the directory where you downloaded it:

```
# tar xzvf MySQL-python-1.2.2.tar.gz
# cd MySQL-python-1.2.2
# python setup.py build
# python setup.py install
```

## Making Application and Template Directories

Django applications aren't deployed like PHP, for example. Your application's source files and templates are kept outside of your web server's document root (much better for security), and you instead use Django's URL resolvers to build a logical URL structure. This also has the disadvantage of meaning you have to put your media files for a project somewhere separately, but I'll cover that later.

For now, we need to make both an application directory, where our application code will reside, and a template directory. In CentOS, the document root by default is `/var/www/html`, meaning we have the whole of `/var/www/` to use for other web server stuff that needs to remain outside the web server's normal root. Perfect for this.

```
# mkdir /var/www/djangoapps
# chown apache /var/www/djangoapps
# mkdir /var/www/djangotemplates
# chown apache /var/www/djangotemplates
```

## Make an Egg Cache

We're almost done in the preparation stage, but we also have to add a Python egg cache that the web server can write to. I found the easiest way to do this was to add a new directory in **/var/cache** for the purpose.

```
# mkdir -p /var/cache/www/pythoneggs
# chown apache /var/cache/www/pythoneggs
```

## Serving up Media Files

Media files have to be done separately. If you're into Vhosts, you could set up a subdomain for serving media, but the simplest way is to just make a `/media` directory in **/var/www/html** and place your Django application's media in a subfolder of that (then updating **settings.py** for each app to reflect the location).

If you're planning on using the Django admin interface, I recommend symlinking the Django admin media directory in here too, like so:

```
# ln -s /opt/django/trunk/django/contrib/admin/media
/var/www/html/media/admin
```

Now, set `ADMIN_MEDIA` to this path in the **settings.py** file for each app you deploy.

## Deploying an Application

Finally, you need to actually deploy an application. Here, I'll assume you're working with an application named **myapp**. Initially, you should drop the application source code in **/var/www/djangoapps/myapp**, the templates in **/var/www/djangotemplates/myapp** and your media in the media location you set up earlier.

Make sure at this point you now go back to **/var/www/djangoapps/myapp/settings.py** and tweak the project's settings, including Debug mode, media URL, admin media URL and anything else relevant to your new deployment. Also, don't forget to check your database settings if you're working with a different database server or instance.

You also need to go into **urls.py** and update the URL patterns with a prefix that you want to use for your application (in this case, **/myapp**), or your links will be broken and URLs won't resolve once installed.


In httpd.conf, add a Location tag to specify your new application's root:

```
<Location /myapp>
SetHandler python-program
PythonHandler django.core.handlers.modpython
SetEnv DJANGO_SETTINGS_MODULE myapp.settings
SetEnv PYTHON_EGG_CACHE "/var/cache/www/pythoneggs"
PythonDebug Off
PythonPath ["'/var/www/djangoapps'"] + sys.path"
</Location>
```

Save httpd.conf, and restart Apache:

```
# /etc/init.d/httpd restart
```

Also remember that any changes to source code will likely require a restart of Apache to read them in.



Peter Upfold

Tips & Tutorials

Server

tutorials

tutorial

Apache

installation

CentOS

Django

mod\_python

web server

HOME » ARTICLES »

© 2006 - 2010 Oratos Media.

FOSSwire is an Oratos Media property. Content is made available under the CC-BY-SA 3.0 license.