**DJANGO IMAGE AND FILE UPLOAD**

This tutorial shows how to implement file and then image uploading with Django.

**SETUP**

On the command line, navigate there and create a directory  djangogirls  for our files. We will use [pip env](https://pipenv.readthedocs.io/en/latest/) to install both Django and [pillow](https://python-pillow.org/) which is a Python image process library Django relies on for image files. For non-image file uploads, pillow is not needed. Finally activate our new virtual environment with the shell command.

* **mkdir djangogirls**
* **cd djangogirls**
* **python –m venv myvenv**
* **myvenv\Scripts\activate**
* **python –m pip install –upgrade pip**

**Installing packages with requirements**

A requirements file keeps a list of dependencies to be installed using pip install:

First create a requirements.txt file inside of the djangogirls/ folder, using the code editor that you installed earlier. You do this by opening a new file in the code editor and then saving it as requirements.txt in the djangogirls/ folder. Your directory will look like this:

djangogirls

├── myvenv

│ └── ...

└───**requirements.txt**

In your djangogirls/requirements.txt file you should add the following text:

djangogirls/requirements.txt

**Django~=2.2.4**

Now run requirements.txt to install django.

* **pip install –r requirements.txt**

## Project and App

* **django-admin.exe startproject mysite .**
* **python manage.py startapp posts**

Since we've added a new app we need to tell Django about it at the bottom of the INSTALLED\_APPS configuration in settings.py.

*# mysite/settings.py*

INSTALLED\_APPS **=** [

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

'posts', *# new*

]

Now run

* **python manage.py migrate**
* **python manage.py runserver**

**Models**

Starting with the database model is a good choice. In our case our model Post will only have two fields: title and cover. We'll also include a \_\_str\_\_ method below so that the title appears in our Django admin later on.

*# posts/models.py*

from django.db import models

**class** **Post**(models**.**Model):

title **=** models**.**TextField()

cover **=** models**.**ImageField(upload\_to**=**'images/')

**def** \_\_str\_\_(self):

**return** self**.**title

The location of the uploaded image will be in MEDIA\_ROOT/images. In Django, the MEDIA\_ROOT setting is where we define the location of all user uploaded items. We'll set that now.If we wanted to use a regular file here the only difference could be to change ImageField to FileField.

Media Root

Open up mysite/settings.py in your text editor. We will add two new configurations. By default MEDIA\_URL and MEDIA\_ROOT are empty and not displayed so we need to configure them.

* MEDIA\_ROOT is the absolute filesystem path to the directory for user-uploaded files
* MEDIA\_URL is the URL we can use in our templates for the fil

***# mysite/settings.py***

**MEDIA\_URL = '/media/'**

**MEDIA\_ROOT = os.path.join(BASE\_DIR, 'media')**

**Admin**

Now update the posts/admin.py file so we can see our Post app in the Django admin.

**#*posts/admin.py***

**django.contrib import admin**

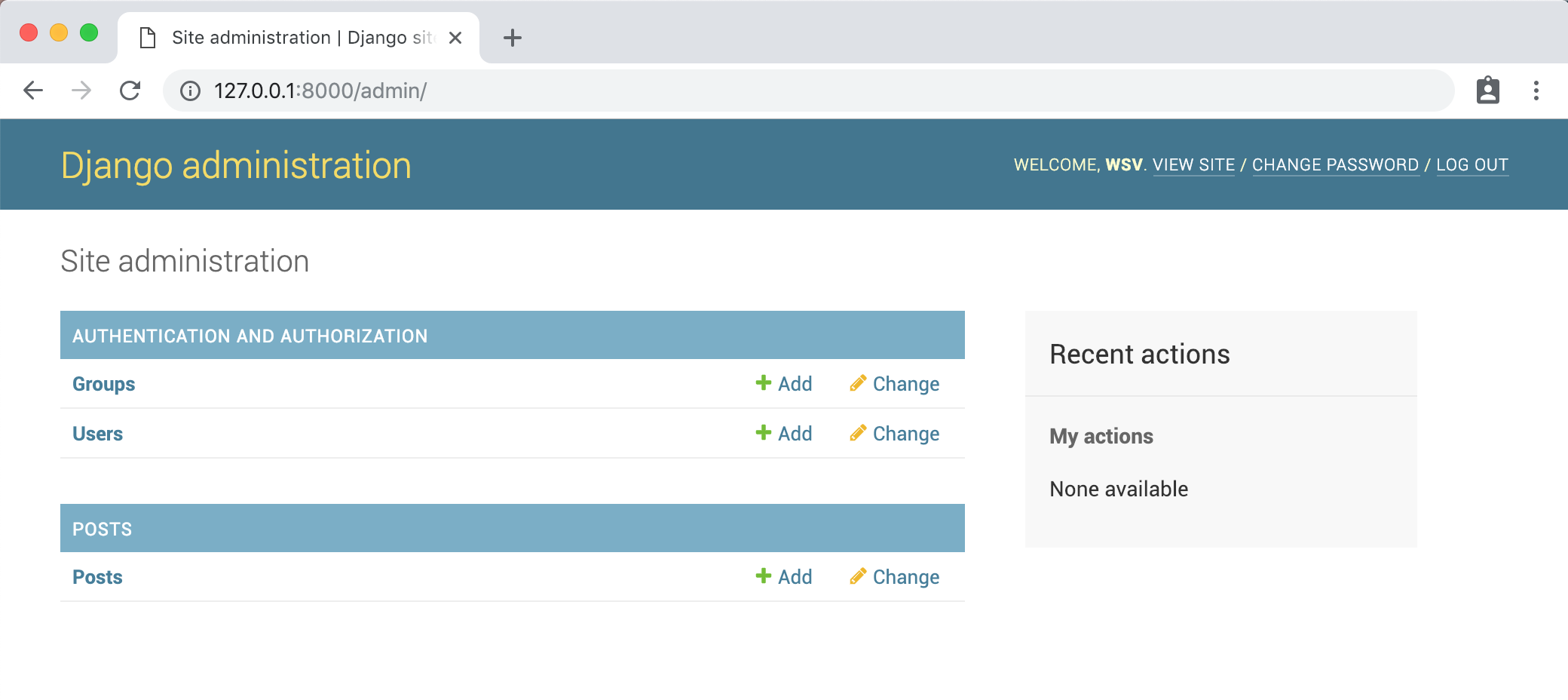
**from .models import Post**

**admin.site.register(Post)**

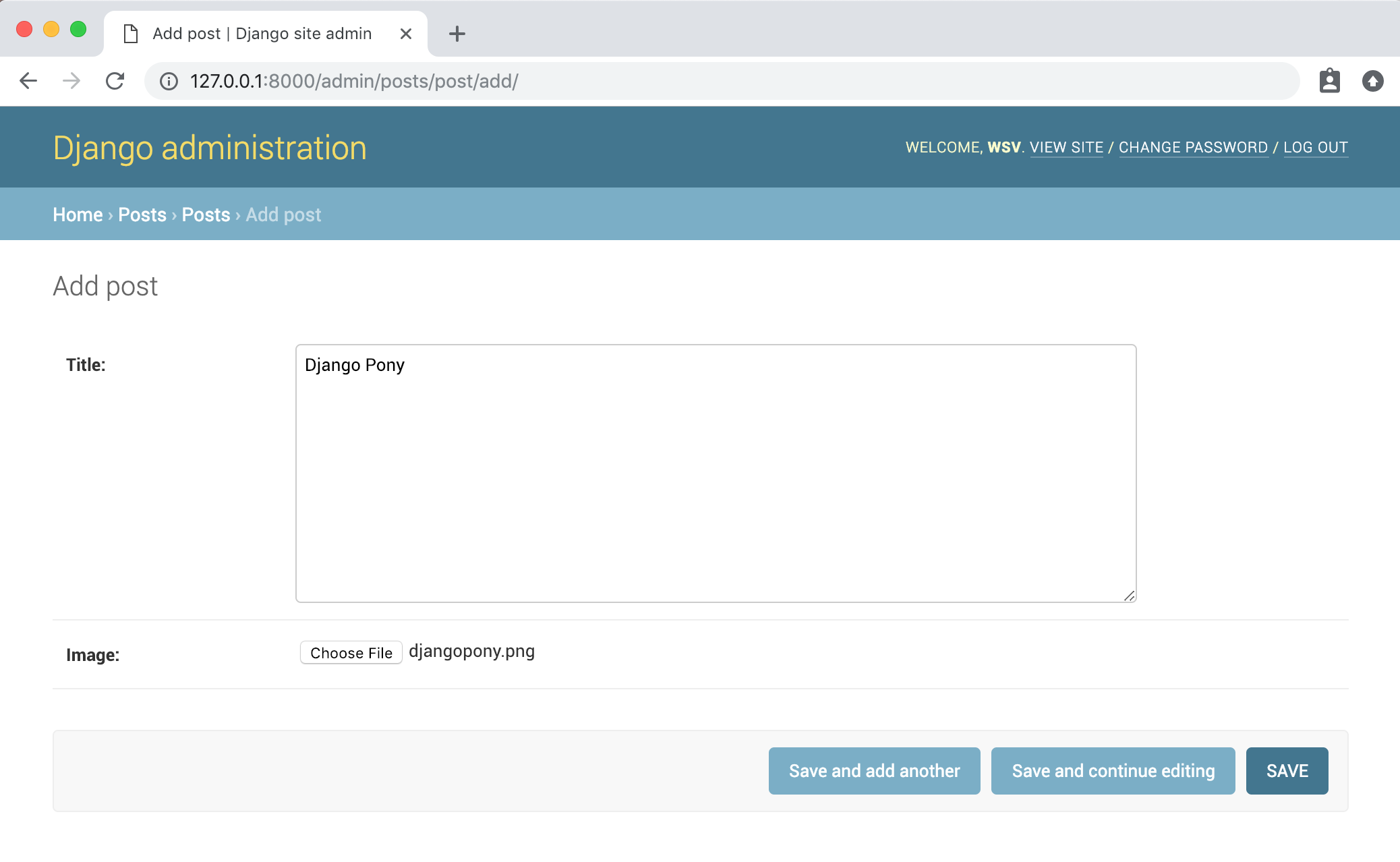
Now run,

* **pip install Pillow**
* **python manage.py makemigrations**
* **python manage.py migrate**
* **python manage.py createsuperuser**
* **python manage.py runserver**

If you go to <http://127.0.0.1:8000/admin> you'll be able to log in to the Django admin site. It should redirect you to this page.



Click on the "+ Add" link next to Posts. You can add whatever you like .



Upon "Save" you will be redirected to the Posts page where we can see all our posts.

If you look within the local media folder in your project you'll see under images there is now the djangopony.jpeg image file.

**Urls**

The confusing thing about Django is that you often need 4 different but interconnected files for one webpage: models.py, urls.py, views.py, and a template html file. I find it easiest to reason about this by going in order from models -> urls -> views -> template files. Our model is already done so that means diving into URL routes.

We'll need two urls.py file updates. First at the project-level config/urls.py files we need to add imports for settings, include, and static. Then define a route for the posts app. Note we also need to add the MEDIA\_URL if settings are in DEBUG mode, otherwise we won't be able to view uploaded images locally.

*# mysite/urls.py*

from django.contrib import admin

from django.conf import settings *# new*

from django.urls import path, include *# new*

from django.conf.urls.static import static *# new*

urlpatterns **=** [

path('admin/', admin**.**site**.**urls),

path('', include('posts.urls')), *# new*

]

**if** settings**.**DEBUG: *# new*

urlpatterns **+=** static(settings**.**MEDIA\_URL, document\_root**=**settings**.**MEDIA\_ROOT)

Next we'll need to sort out the URL routes within the posts app. First create that file.

*# posts/urls.py*

from django.urls import path

from .views import HomePageView

urlpatterns **=** [

path('', HomePageView**.**as\_view(), name**=**'home'),

]

This references a view called HomePageView which we'll create next.

**Views**

We can use the generic class-based ListView here, import our Post model, and then create a HomePageView that uses the model and a template called home.html.

*# posts/views.py*

from django.views.generic import ListView

from .models import Post

**class** **HomePageView**(ListView):

model **=** Post

template\_name **=** 'home.html'

**Templates**

We could put it within the posts app at posts/templates/posts/home.html

We tell Django to also look here for any templates by updating the TEMPLATES configuration within mysite/settings.py.

TEMPLATES **=** [

{

**...**

'DIRS': [os.path.join(BASE\_DIR,‘posts’,’templates',’posts’)], *# new*

**...**

},

]

*#posts/templates/posts/home.html -->*

<h1>Django Image Uploading</h1>

<ul>

{% for post in object\_list %}

<h2>{{ post.title }}</h2>

<img src**=**"{{ post.cover.url}}" alt**=**"{{ post.title }}">

{% endfor %}

</ul>

Make sure the server is running with the python manage.py runserver command and navigate to our homepage at [http://127.0.0.1:8000](http://127.0.0.1:8000/).

**Forms**

Now we can add a form so regular users, who wouldn't have access to the admin, can also add posts. That means creating a new page with a form.

Let's start with the views.py file. We'll name our new view CreatePostView which will extend the built-in Django [CreateView](https://docs.djangoproject.com/en/3.1/ref/class-based-views/generic-editing/#django.views.generic.edit.CreateView). We'll also import [reverse\_lazy](https://docs.djangoproject.com/en/3.1/ref/urlresolvers/#reverse-lazy) to handle the redirect back to our homepage after the form has been submitted.

Within the view we specify the model, a form\_class which we'll create next, the template\_name, and finally a success\_url which is what we want to happen after submission.

*# posts/views.py*

from django.views.generic import ListView, CreateView *# new*

from django.urls import reverse\_lazy *# new*

from .forms import PostForm *# new*

from .models import Post

**class** **HomePageView**(ListView):

model **=** Post

template\_name **=** 'home.html'

**class** **CreatePostView**(CreateView): *# new*

model **=** Post

form\_class **=** PostForm

template\_name **=** 'post.html'

success\_url **=** reverse\_lazy('home')

Next up that form. First create it.

posts/forms.py

We can extend Django's built-in ModelForm. All we need for our basic form is to specify the correct model Post and the fields we want displayed which are title and cover.

*# posts/forms.py*

from django import forms

from .models import Post

**class** **PostForm**(forms**.**ModelForm):

**class** **Meta**:

model **=** Post

fields **=** ['title', 'cover']

We'll make a dedicated page for this form at the path of post/.

*# posts/urls.py*

from django.urls import path

from .views import HomePageView, CreatePostView *# new*

urlpatterns **=** [

path('', HomePageView**.**as\_view(), name**=**'home'),

path('post/', CreatePostView**.**as\_view(), name**=**'add\_post') *# new*

]

Then create the new template.

And fill it with a headline and form. It's important to always add csrf\_token for protection. We're specifying form.as\_p which means Django will output each field as a paragraph tag.

*<posts/templates/posts/post.html -->*

<h1>Create Post Page</h1>

<form method**=**"post" enctype**=**"multipart/form-data">

{% csrf\_token %}

{{ form.as\_p }}

<button type**=**"submit">Submit New Post</button>

</form>

That's it! Make sure your server is running and go to the page at <http://127.0.0.1:8000/post/>.