

Topic: Image Compression 26: Thumbnail, Size, Automatic Download

Speaker: Udemy Instructor Rathan Kumar | Notebook: Django: Automating Common Tasks



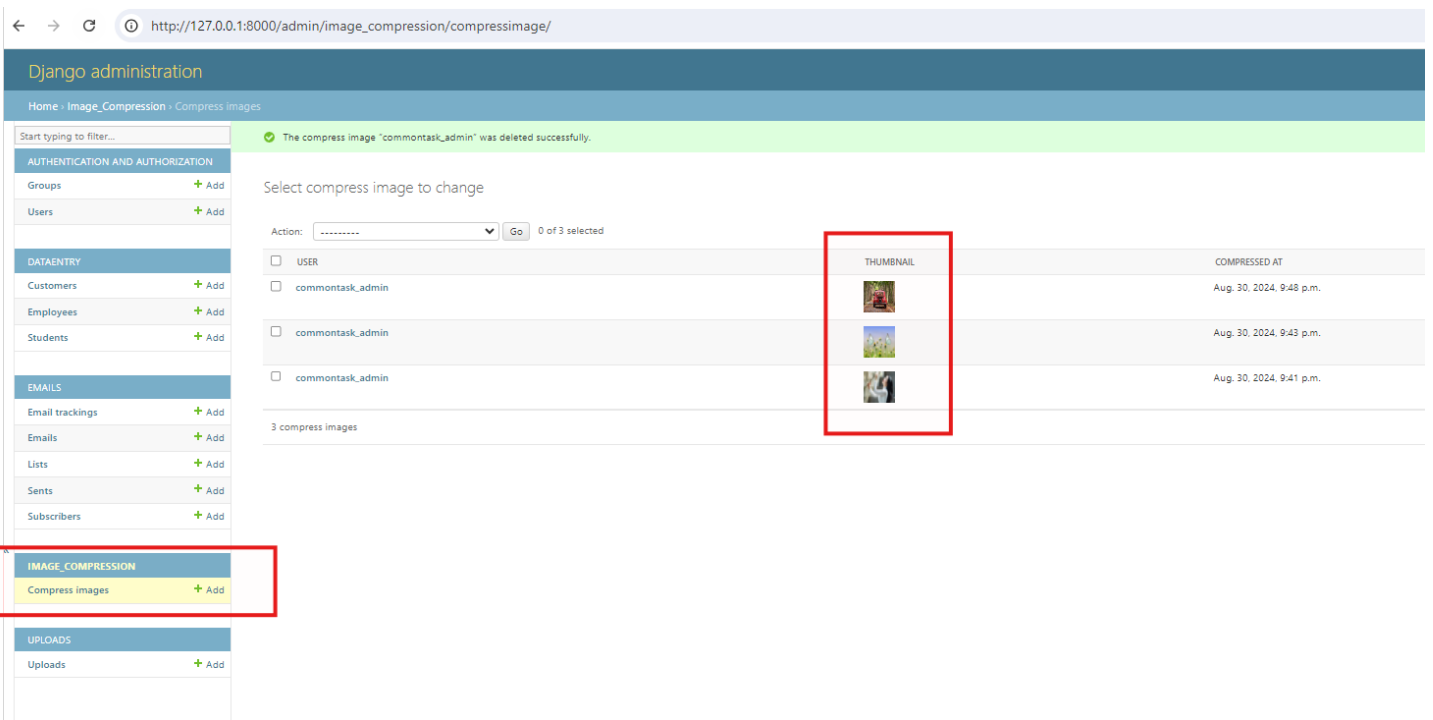
1. We need some thumbnails for our images on the admin dashboard. So, update the ADMIN.PY as:

```
AutomatingCommonTasks

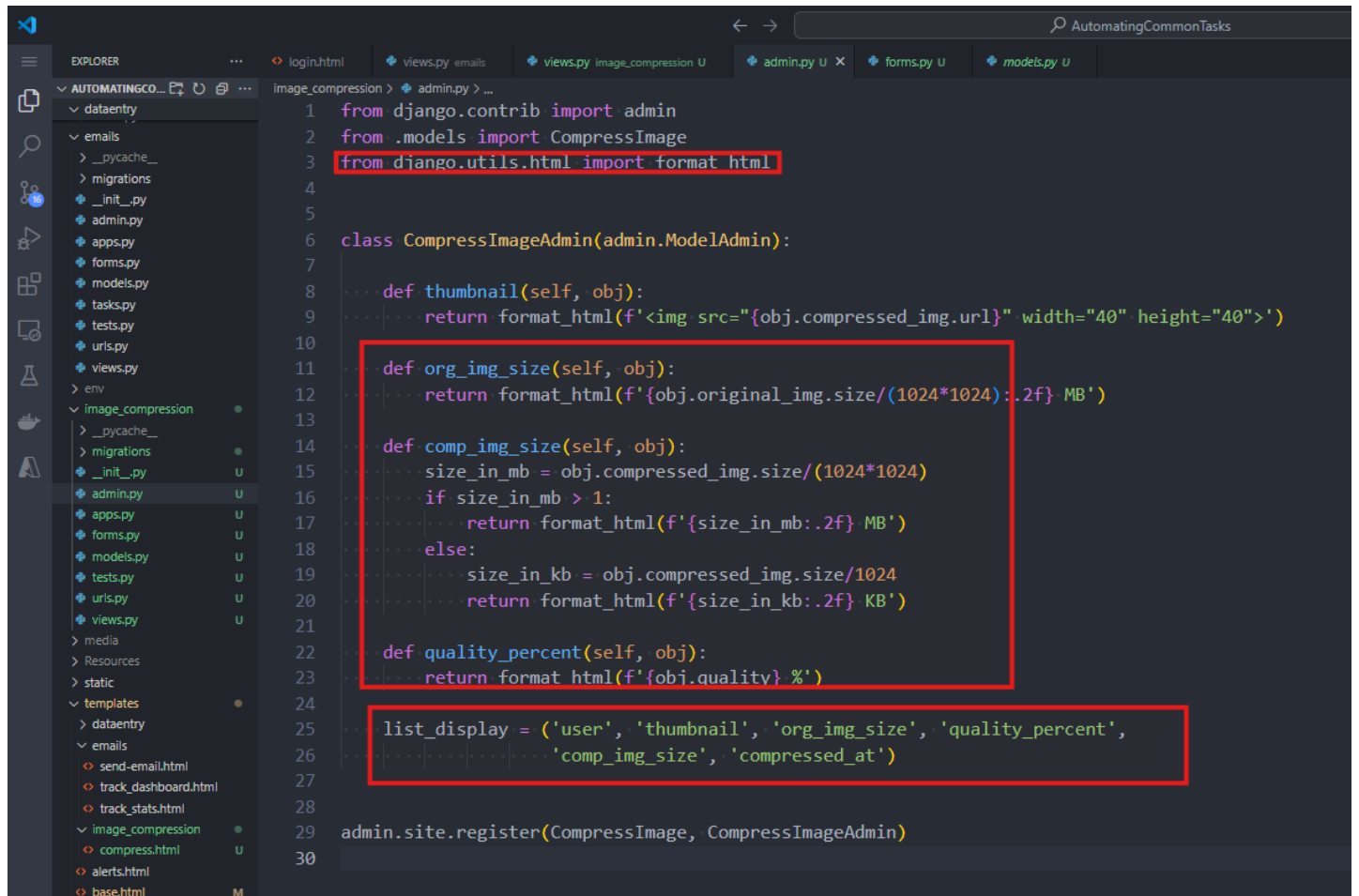
EXPLORER
AUTOMATINGCOMMONTASKS
  autocommontasks_main
    wsgi.py
    dataentry
      __pycache__
      management
      migrations
      __init__.py
      admin.py
      apps.py
      models.py
      tasks.py
      tests.py
      uris.py
      utils.py
      views.py
    emails
      __pycache__
      migrations
      __init__.py
      admin.py
      apps.py
      forms.py
      models.py
      tasks.py
      tests.py
      uris.py
      views.py
    env
    image_compression
      __pycache__
      migrations
      __init__.py
      admin.py
      apps.py
      forms.py

image_compression > admin.py > ...
1 from django.contrib import admin
2 from .models import CompressImage
3 from django.utils.html import format_html
4
5
6 class CompressImageAdmin(admin.ModelAdmin):
7     def thumbnail(self, obj):
8         return format_html(f'')
9
10    list_display = ('user', 'thumbnail', 'compressed_at')
11
12
13 admin.site.register(CompressImage, CompressImageAdmin)
14
```

2. Reload your admin panel:



3. To display the original and compressed sizes of the image, in the ADMIN.PY, we update:



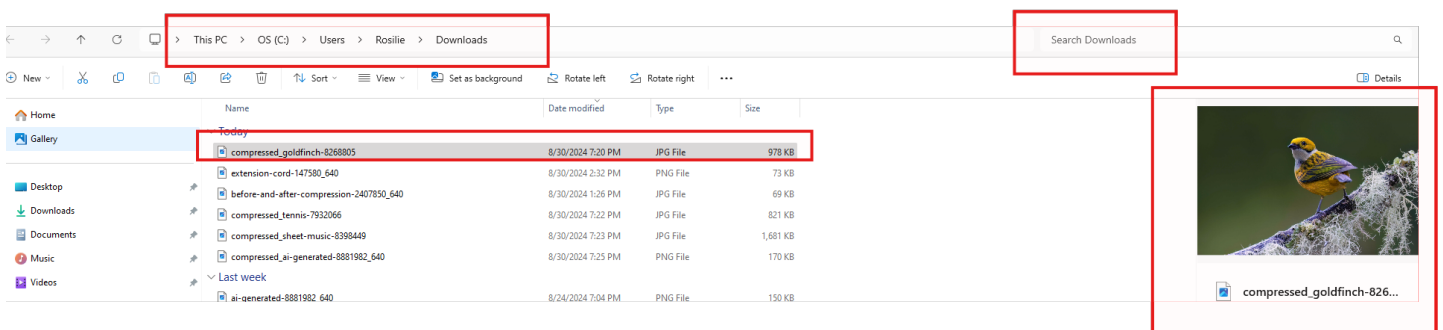
4. To allow the user to download automatically the compressed image, we update our VIEWS.PY

```

AUTOMATINGCOMMONTASKS AutomatingCommonTasks
EXPLORER login.html views.py emails views.py image_compression U admin.py U forms.py U models.py U
image_compression > views.py > compress
dataentry
  urls.py
  utils.py
  views.py
emails
  __pycache__
  migrations
  __init__.py
  admin.py
  apps.py
  forms.py
  models.py
  tasks.py
  tests.py
  urls.py
  views.py
env
image_compression
  __pycache__
  migrations
  __init__.py
  admin.py
  apps.py
  forms.py
  models.py
  tasks.py
  tests.py
  urls.py
  views.py
media
Resources
static
templates
  dataentry
  emails
  send_email.html
  track_dashboard.html
  track_stats.html
  image_compression
  compress.html
  alerts.html
  base.html
  home.html
  login.html
  register.html
  uploads
  .env
  .gitignore
  db.sqlite3
  manage.py
  requirements.txt
OUTLINE
  redirect def redirect
  render def render
  CompressImageForm class ...
  Image module Image
  io module io
  messages module messages
  HttpResponseRedirect class HttpRes...
  compress def compress
2 from .forms import CompressImageForm
3 from PIL import Image
4 import io
5 from django.contrib import messages
6 from django.http import HttpResponseRedirect
7
8
9 def compress(request):
10     user = request.user
11     if request.method == 'POST':
12         form = CompressImageForm(request.POST, request.FILES)
13         if form.is_valid():
14             original_img = form.cleaned_data['original_img']
15             quality = form.cleaned_data['quality']
16             # temporarily saves the form
17             compressed_image = form.save(commit=False)
18             compressed_image.user = user
19             # perform the compression
20             img = Image.open(original_img)
21             # set the image format based on the uploaded image's format
22             output_format = img.format
23             buffer = io.BytesIO()
24             img.save(buffer, format=output_format, quality=quality)
25             buffer.seek(0)
26             # save teh compressed image inside the model with filename format
27             compressed_image.compressed_img.save(
28                 f'compressed_{original_img}', buffer
29             )
30             # return redirect('compress')
31             # Automatically download the compressed file not as binary value but as formatted image
32             response = HttpResponseRedirect(
33                 buffer.getvalue(), content_type=f'image/{output_format.lower()}'
34             )
35             response['Content-Disposition'] = f'attachment;filename=compressed_{original_img}'
36             return response
37         else:
38             form = CompressImageForm()
39             context = {
40                 'form': form,
41             }
42             return render(request, 'image_compression/compress.html', context)

```

5. Open your downloads folder for the images:



6. Push your changes to Github.