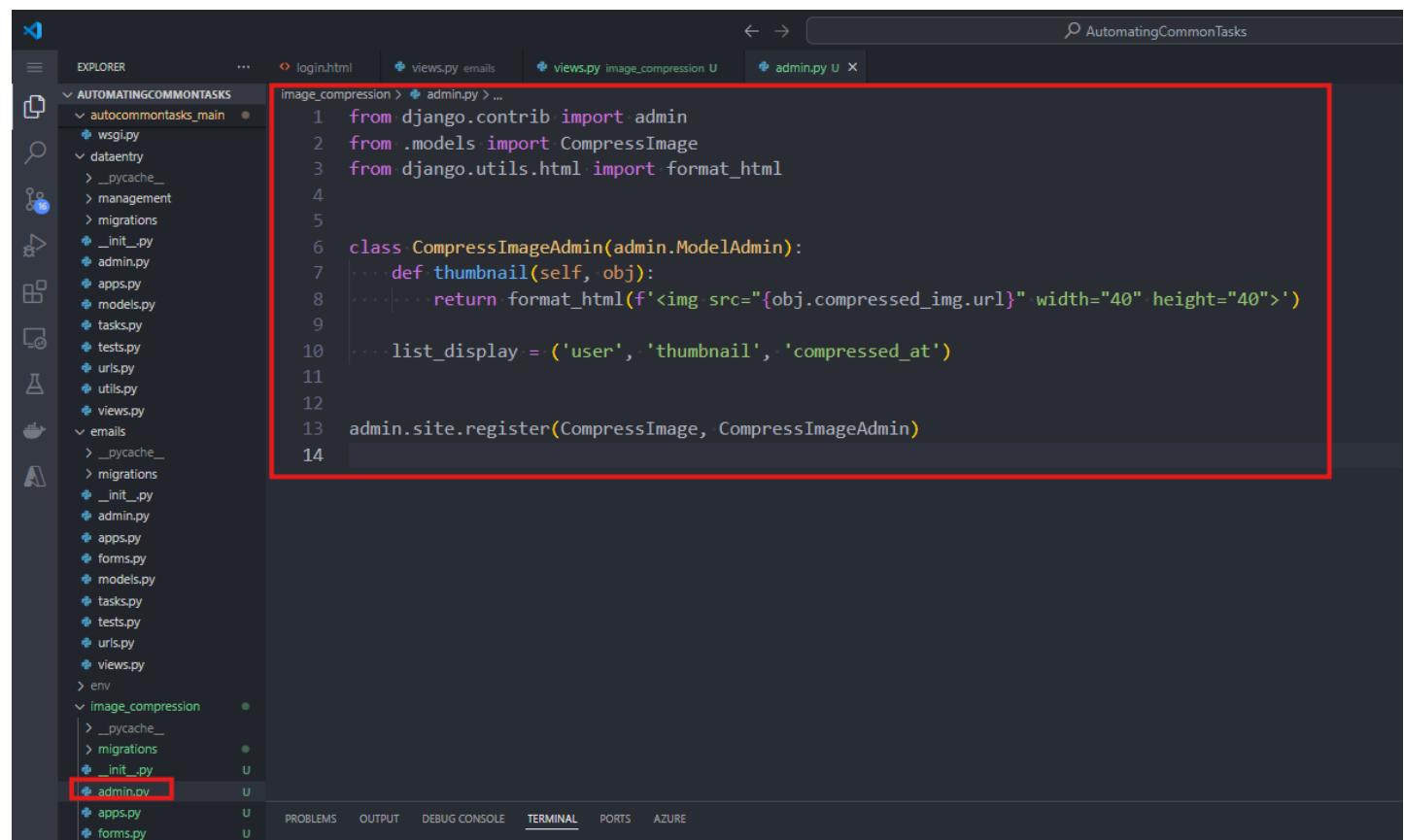


## Topic: Image Compression 26: Thumbnail, Size, Automatic Download

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:



```
from django.contrib import admin
from .models import CompressImage
from django.utils.html import format_html

class CompressImageAdmin(admin.ModelAdmin):
    def thumbnail(self, obj):
        return format_html(f'')

    list_display = ('user', 'thumbnail', 'compressed_at')

admin.site.register(CompressImage, CompressImageAdmin)
```

The code editor shows the `admin.py` file with a red box highlighting the `thumbnail` method. The `admin` folder is also highlighted with a red box.

2. Reload your admin panel:

Django administration

Home > Image\_Compression > Compress images

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION

- Groups [+ Add](#)
- Users [+ Add](#)

DATAENTRY

- Customers [+ Add](#)
- Employees [+ Add](#)
- Students [+ Add](#)

EMAILS

- Email trackings [+ Add](#)
- Emails [+ Add](#)
- Lists [+ Add](#)
- Sents [+ Add](#)
- Subscribers [+ Add](#)

**IMAGE\_COMPRESSION**

Compress images [+ Add](#)

UPLOADS

- Uploads [+ Add](#)

Select compress image to change

Action:   0 of 3 selected

	USER	THUMBNAIL	COMPRESSED AT
<input type="checkbox"/>	commenttask_admin		Aug. 30, 2024, 9:48 p.m.
<input type="checkbox"/>	commenttask_admin		Aug. 30, 2024, 9:43 p.m.
<input type="checkbox"/>	commenttask_admin		Aug. 30, 2024, 9:41 p.m.

3 compress images

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

AutomatingCommonTasks

EXPLORER

- AUTOMATINGCO... [+ Add](#)
- dataentry
- emails
- migrations
- image\_compression**
  - \_\_pycache\_\_
  - migrations
  - \_\_init\_\_.py**
  - admin.py
  - apps.py
  - forms.py
  - models.py
  - tasks.py
  - tests.py
  - urls.py
  - views.py
- env
- media
- Resources
- static
- templates
- dataentry
- emails
- image\_compression
- compress.html
- alerts.html
- base.html

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
8      def thumbnail(self, obj):
9          return format_html(f'')
10
11     def org_img_size(self, obj):
12         return format_html(f'{obj.original_img.size/(1024*1024):.2f} MB')
13
14     def comp_img_size(self, obj):
15         size_in_mb = obj.compressed_img.size/(1024*1024)
16         if size_in_mb > 1:
17             return format_html(f'{size_in_mb:.2f} MB')
18         else:
19             size_in_kb = obj.compressed_img.size/1024
20             return format_html(f'{size_in_kb:.2f} KB')
21
22     def quality_percent(self, obj):
23         return format_html(f'{obj.quality}%')
24
25     list_display = ('user', 'thumbnail', 'org_img_size', 'quality_percent',
26                    'comp_img_size', 'compressed_at')
27
28
29 admin.site.register(CompressImage, CompressImageAdmin)
30

```

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

AutomatingCommonTasks

File Explorer

image\_compression > views.py > 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

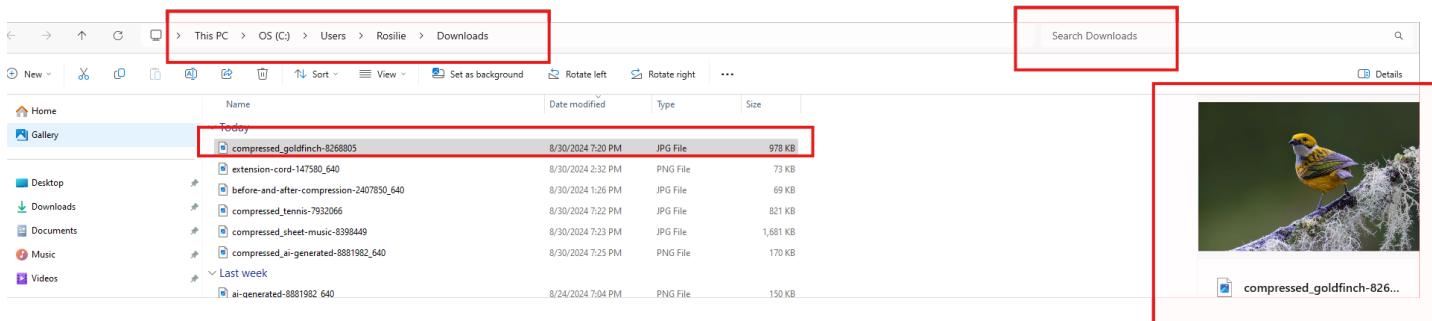
7
8
9  def compress(request):
10     user = request.user
11     if request.method == 'POST':
12         form = CompressImageForm(request.POST, request.FILES)
13         if form.is_valid():
14             original_img = form.cleaned_data['original_img']
15             quality = form.cleaned_data['quality']
16
17             # temporarily saves the form
18             compressed_image = form.save(commit=False)
19             compressed_image.user = user
20
21             # perform the compression
22             img = Image.open(original_img)
23
24             # set the image format based on the uploaded image's format
25             output_format = img.format
26
27             buffer = io.BytesIO()
28             img.save(buffer, format=output_format, quality=quality)
29             buffer.seek(0)
30
31             # save the compressed image inside the model with filename format
32             compressed_image.compressed_img.save(
33                 f'compressed_{original_img}', buffer
34             )
35             # return redirect('compress')
36             # Automatically download the compressed file not as binary value but as formatted image
37             response = HttpResponseRedirect(
38                 buffer.getvalue(), content_type=f'image/{output_format.lower()}')
39             response['Content-Disposition'] = f'attachment;filename=compressed_{original_img}'
40             return response
41
42     else:
43         form = CompressImageForm()
44         context = {
45             'form': form,
46         }
47         return render(request, 'image_compression/compress.html', context)

```

Outline

- redirect
- render
- CompressImageForm
- Image
- io
- messages
- HttpResponse
- compress

5. Open your downloads folder for the images:



6. Push your changes to Github.