

Topic: Image Compression 25: Setup, Model & Logic

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



Overview

The Python Imaging Library adds image processing capabilities to your Python interpreter.

This library provides extensive file format support, an efficient internal representation, and fairly powerful image processing capabilities.

The core image library is designed for fast access to data stored in a few basic pixel formats. It should provide a solid foundation for a general image processing tool.

1. Go to [Pillow documentation](#) and install the Django package in the Django server terminal and in Celery terminal

```
$ pip install pillow
```

2. In the Django server, we create a new app IMAGE_COMPRESSION

```
$ python manage.py startapp image_compression
```

3. Register the new app in SETTINGS.PY INSTALLED_APPS

```

# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'dataentry',
    'uploads',
    'crispy_forms',
    'crispy_bootstrap5',
    'emails',
    'ckeditor',
    'anymail',
    'image_compression',
]

```

4. Create the model in MODELS.PY

```

1 from django.db import models
2 from django.contrib.auth.models import User
3
4
5 class CompressImage(models.Model):
6     # dropdown selection as [(10,10), (20,20), (30,30), ... (100,100)]
7     QUALITY_CHOICES = [(i, i) for i in range(10, 101, 10)]
8
9     user = models.ForeignKey(User, on_delete=models.CASCADE)
10    original_img = models.ImageField(upload_to='original_images/')
11    quality = models.IntegerField(choices=QUALITY_CHOICES, default=80)
12    compressed_img = models.ImageField(upload_to='compressed_images/')
13    compressed_at = models.DateTimeField(auto_now_add=True)
14
15    def __str__(self):
16        return self.user.username
17

```

To make the QUALITY field a drop-down option, we use the SELECT TAG. Where we see the VALUE AND THE LABEL (which is displayed as a user option in the dropdown)

```

<!DOCTYPE html>
<html>
<body>

<h1>The select element</h1>

<p>The select element is used to create a drop-down list.</p>

<form action="/action_page.php">
  <label for="cars">Choose a car:</label>
  <select name="cars" id="cars">
    <option value="volvo">Volvo</option>
    <option value="saab">Saab</option>
    <option value="opel">Opel</option>
    <option value="audi">Audi</option>
  </select>
  <br><br>
  <input type="submit" value="Submit">
</form>

<p>Click the "Submit" button and the form-data will be sent to a page on the
server called "action_page.php".</p>

</body>
</html>

```

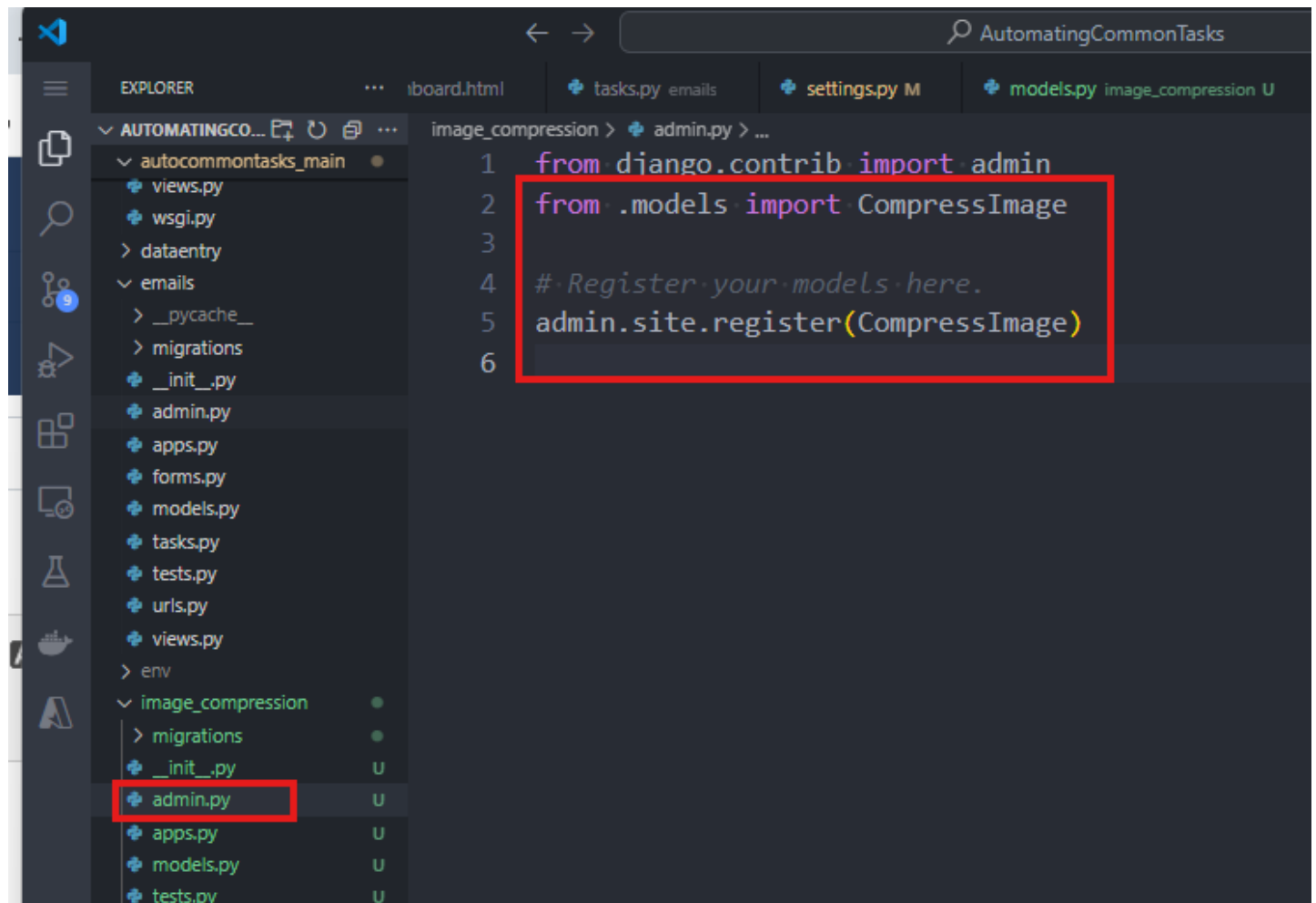
The select element

The select element is used to create a drop-down list.

Choose a car:

Click the "Submit" button and the form-data will be sent to a page on the server

5. Register the model for our ADMIN panel. Update ADMIN.PY:



6. Make the necessary model migrations

```
$ python manage.py makemigrations
```

```
$ python manage.py migrate
```

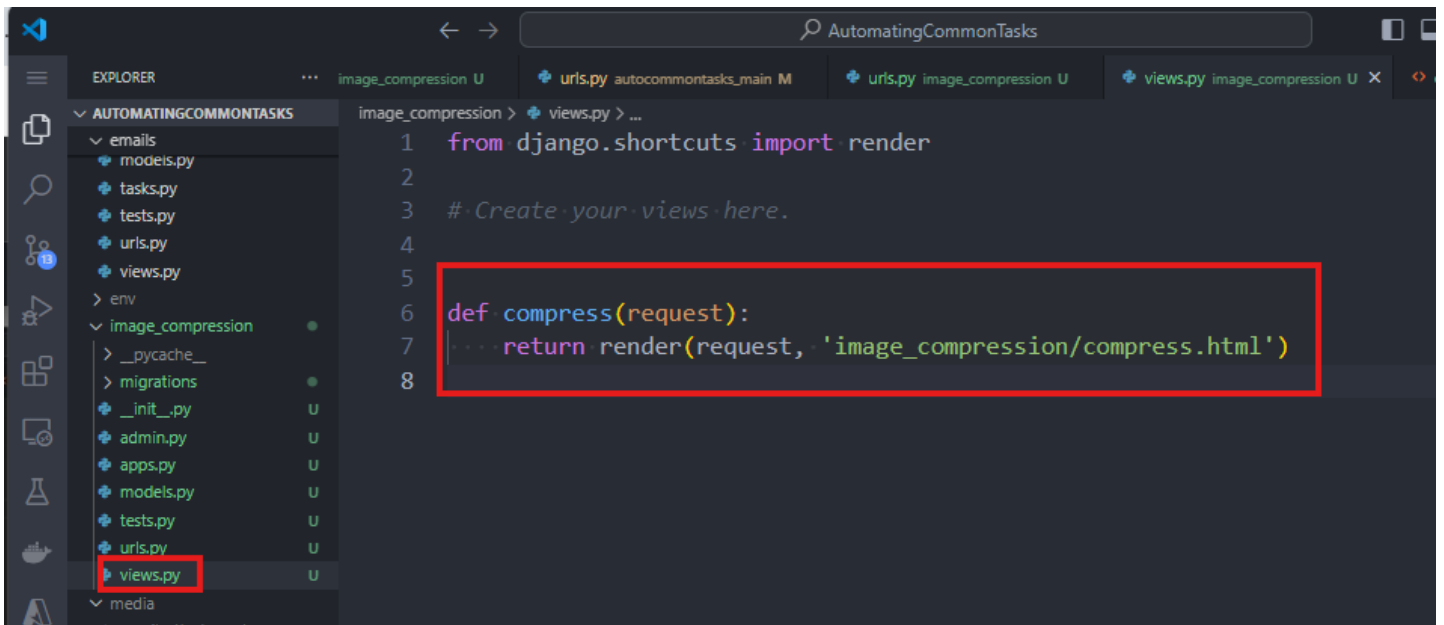
7. Create the new URL pattern. Since we have a new app, we need to create a new pattern in our main project's URLS.PY.

```
10 Class-based views
11 ...1. Add an import: from other_app.views import Home
12 ...2. Add a URL to urlpatterns: path('', Home.as_view(), name='home')
13 Including another URLconf
14 ...1. Import the include() function: from django.urls import include, path
15 ...2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
16 """
17 from django.contrib import admin
18 from django.urls import path, include
19 from . import views
20 from django.conf.urls.static import static
21 from django.conf import settings
22
23 urlpatterns = [
24     path('admin/', admin.site.urls),
25     path('', views.home, name='home'),
26     # links to our dataentry app's URLS.PY
27     path('dataentry/', include('dataentry.urls')),
28     path('celery-test/', views.celery_test),
29     # registration and login
30     path('register/', views.register, name='register'),
31     path('login/', views.login, name='login'),
32     path('logout/', views.logout, name='logout'),
33     # Email tasks to forward to emails app's urls.py
34     path('emails/', include('emails.urls')),
35     # Image compression tasks
36     path('image-compression/', include('image_compression.urls')),
37 ] + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
38
39
```

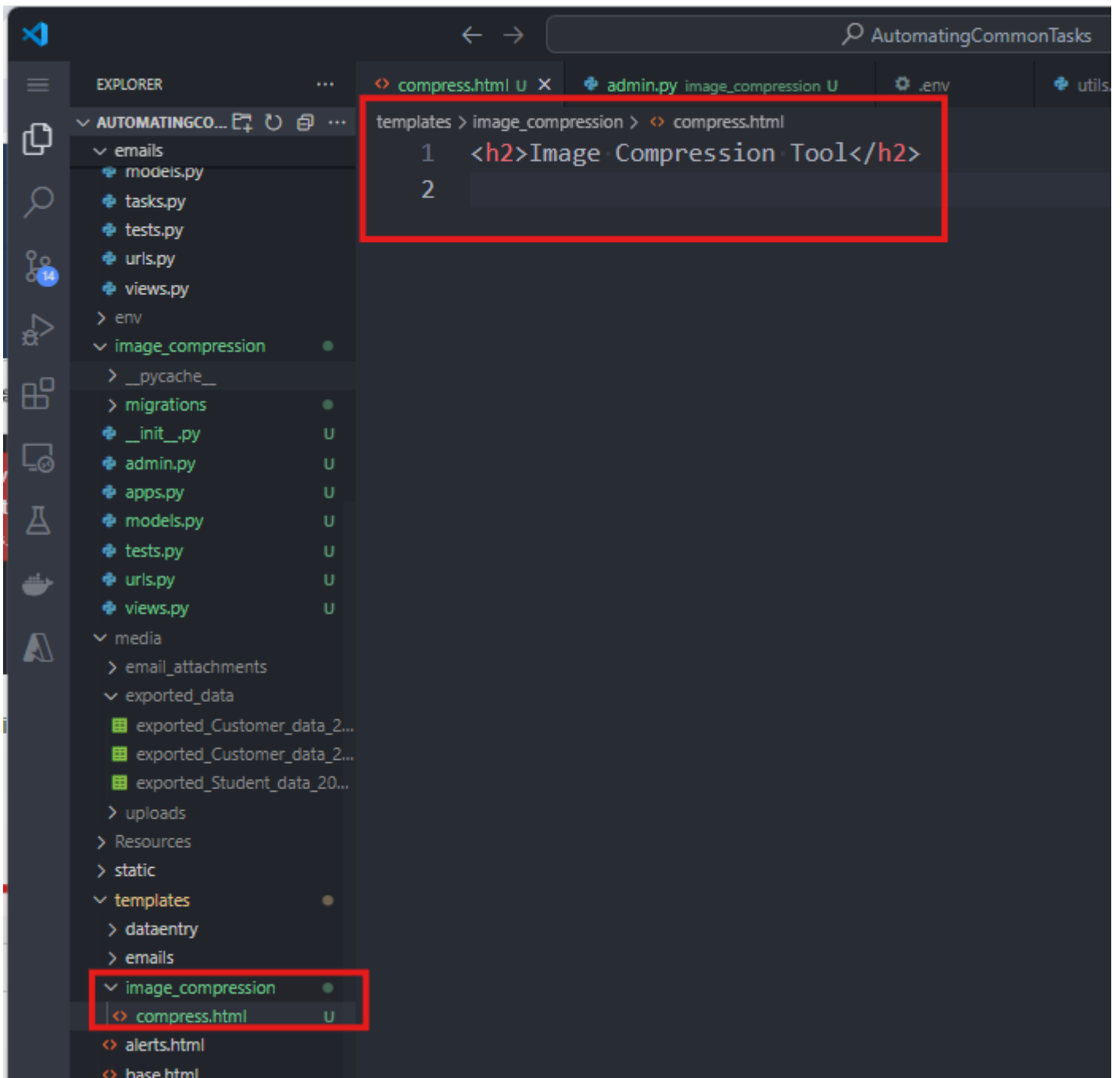
Then, create a new URLS.PY file in our new app for image-compression-related URL paths.

```
1 from django.urls import path
2 from . import views
3
4 urlpatterns = [
5     path('compress/', views.compress, name='compress'),
6 ]
7
8
```

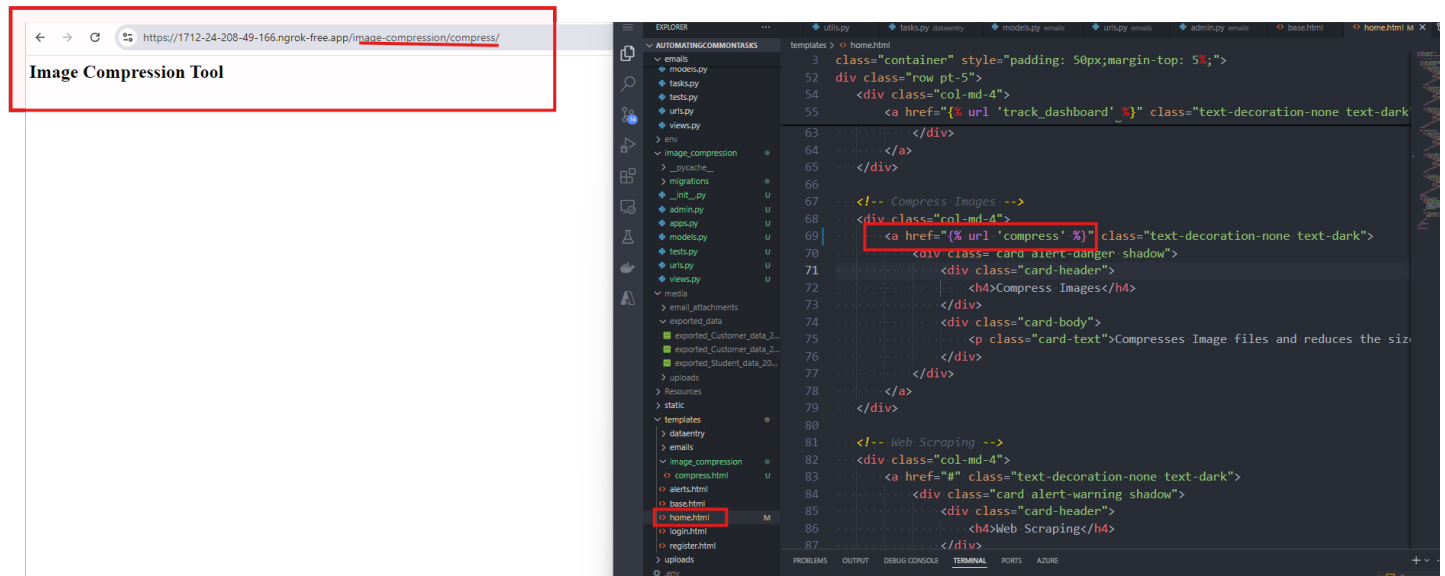
8. Create the function in the VIEWS.PY



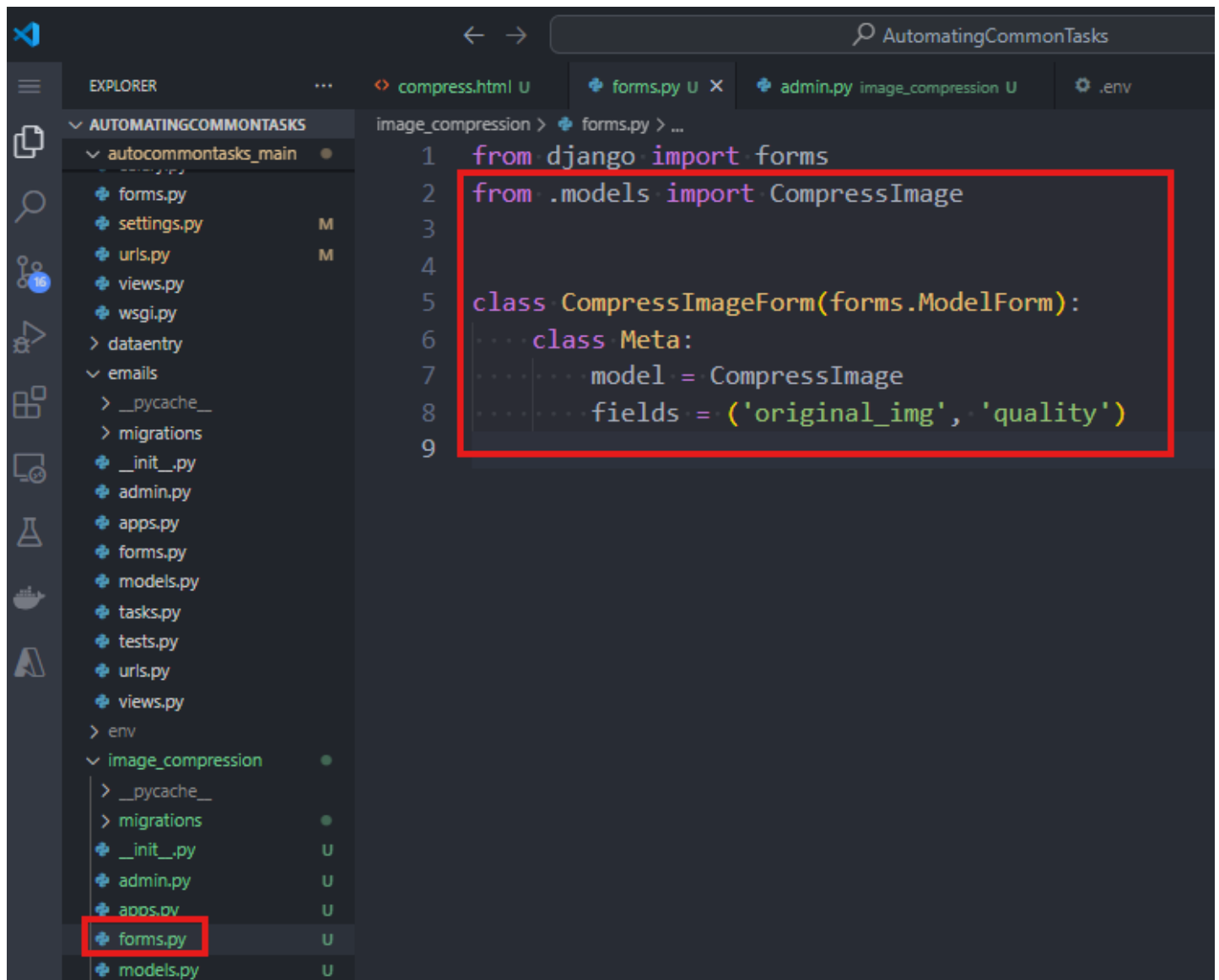
9. Create a new FOLDER called IMAGE_COMPRESSION, and in this folder, create a new file, COMPRESS.HTML



10. Update the HOME.HTML to call this new web page.



11. Create a FORMS.PY



12. Call our form using our VIEWS.PY:

```
1 from django.shortcuts import render
2 from .forms import CompressImageForm
3 # Create your views here.
4
5
6 def compress(request):
7     form = CompressImageForm()
8     context = {
9         'form': form,
10    }
11    return render(request, 'image_compression/compress.html', context)
12
```

13. In the COMPRESS.HTML, update as:

Automate The Common Boring Stuffs with Django

Logged in as: Commentask_admin Logout

Compress Images Tool

Original img*

Choose File No file chosen

Quality*

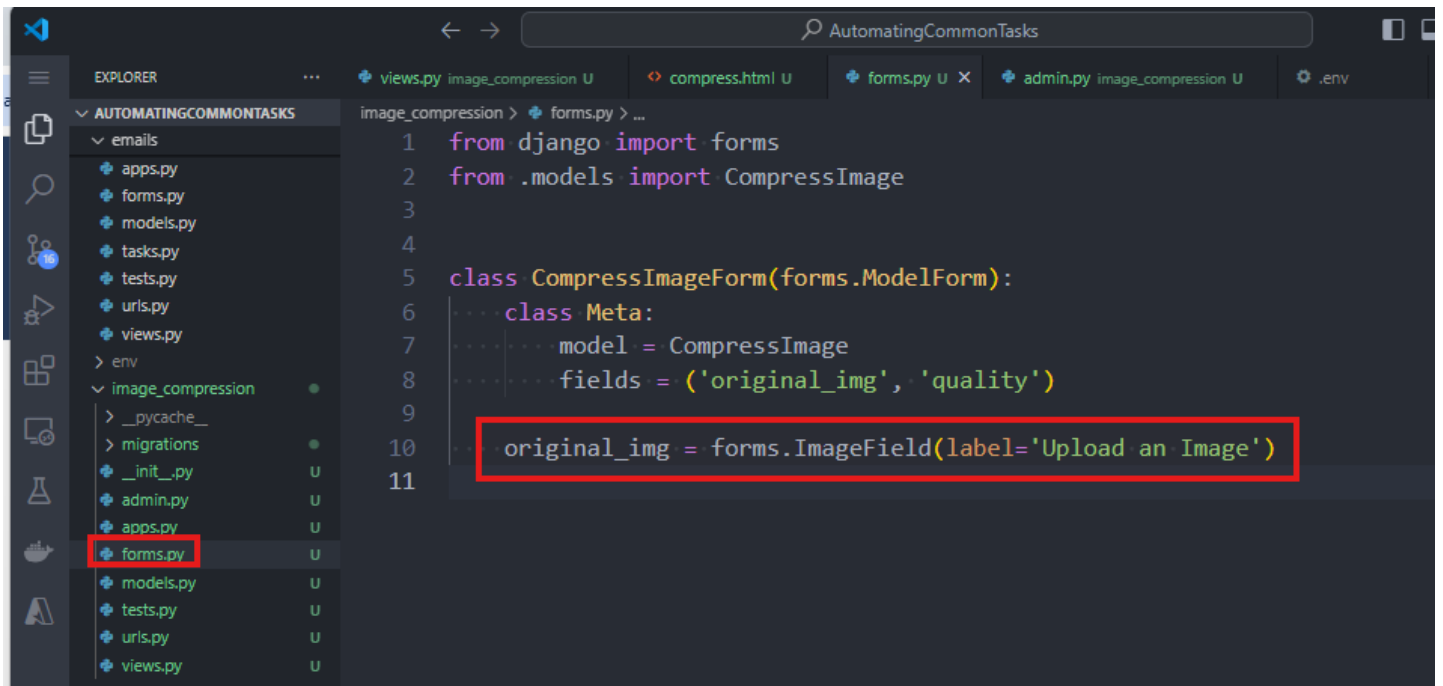
80

Compress

Copyright © 2024 Automate The Common Boring Stuffs | By Rosilie Lim

```
1 {% extends 'base.html' %}
2
3 {% block content %}
4 {% load crispy_forms_tags %}
5
6 <div class="container mt-5 p-3 shadow rounded" style="max-width: 700px;">
7     <h2 class="text-center">Compress Images Tool </h2>
8     <form action="{% url 'compress' %}" method="POST" enctype="multipart/form-data">
9         {{ csrf_token }}
10
11         <div class="form-group">
12             {{ form | crispy }}
13         </div>
14
15         <div class="form-group">
16             <input type="submit" value="Compress" class="btn btn-primary">
17         </div>
18     </form>
19     {% include 'alerts.html' %}
20 </div>
21
22
23 {% endblock %}
```

14. We can change the label of the field on the form, so update FORMS.PY and add the line:

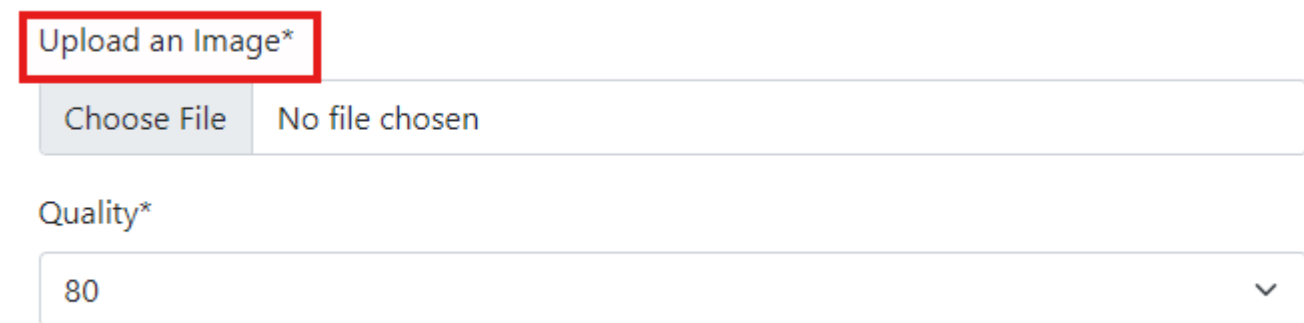


```
1 from django import forms
2 from .models import CompressImage
3
4
5 class CompressImageForm(forms.ModelForm):
6     class Meta:
7         model = CompressImage
8         fields = ('original_img', 'quality')
9
10    original_img = forms.ImageField(label='Upload an Image')
11
```

FROM:



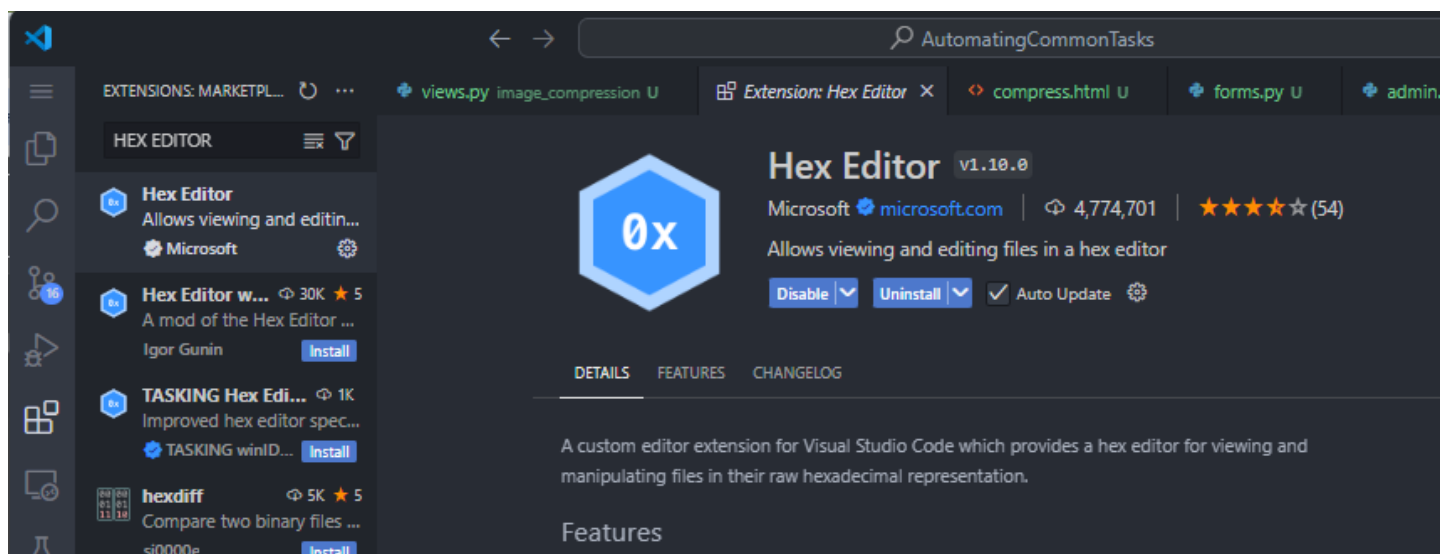
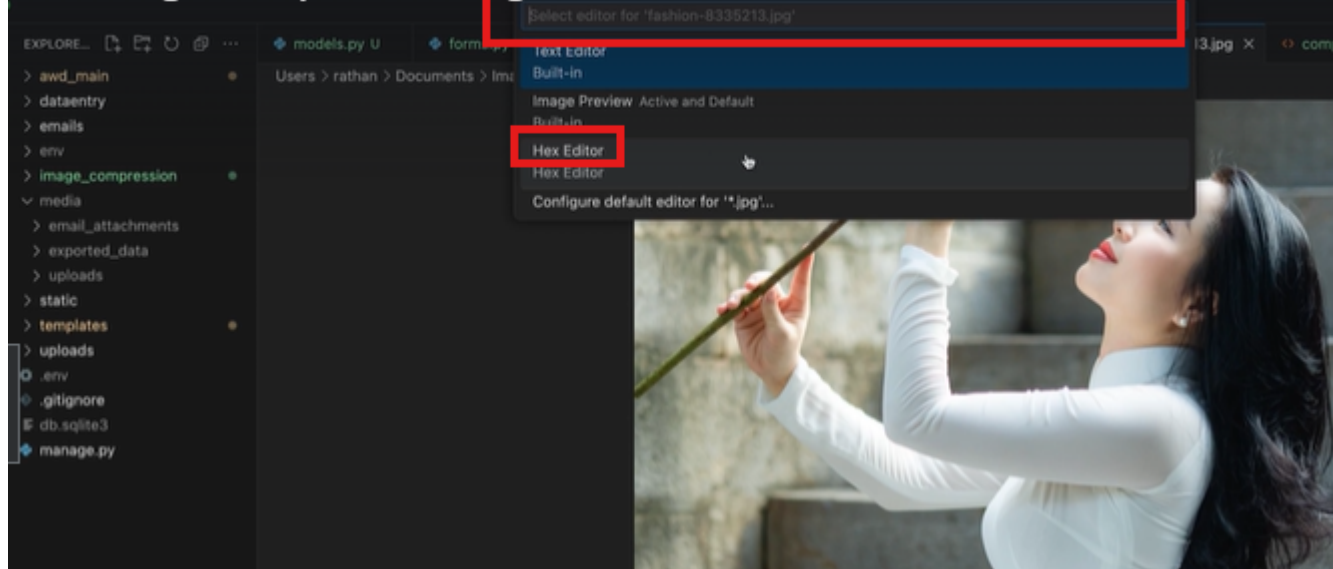
TO:



15. When we use `io.BytesIO`, we get bytes value of the image. To see what is the visual representation of these bytes, you can add the EXTENSION 'HEX EDITOR'.

Now open an image file. On the tab of this image, right-click, select 'REOPEN EDITOR WITH', and select HEXEDITOR and you will see the HEX value of the file.

103. Image Compression Logic




```
7 def compress(request):
17
18     ##### perform the compression
19     img = Image.open(original_img)
20     buffer = io.BytesIO()
21     print('buffer cursor position or pointer at the beginning = >', buffer.tell())
22
23     img.save(buffer, format='JPEG', quality=quality)
24     ##### print('buffer:=>', buffer.getvalue())
25     print('buffer cursor position or pointer after image compression = >', buffer.tell())
26     ##### pointer goes back to 0 location after saving in the buffer
27     buffer.seek(0)
28     print('buffer cursor position or pointer after return to zero = >', buffer.tell())
29
30     ##### save teh compressed image inside the model with filename format
31     compressed_image.compressed_img.save(
32         f'compressed_{original_img}', buffer
33     )
34     return redirect('compress')
35
36 else:
37     form = CompressImageForm()
38     context = {
39         'form': form,
40     }
41     return render(request, 'image_compression/compress.html', context)
42
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS AZURE

?: (ckeditor.W001) django-ckeditor bundles CKEditor 4.22.1 which isn't supported anymore and which does have unfixed security issues, see for example <https://ckeditor.com/cke4/release>, or switch to the non-free CKEditor 5 LTS package. See <https://ckeditor.com/cke5/migration> for more details. Notice has been added by the django-ckeditor developers and we are not affiliated with CKSource and were not involved in the licensing change, so please refrain from complaining to us.

System check identified 1 issue (0 silenced).

August 30, 2024 - 16:35:55

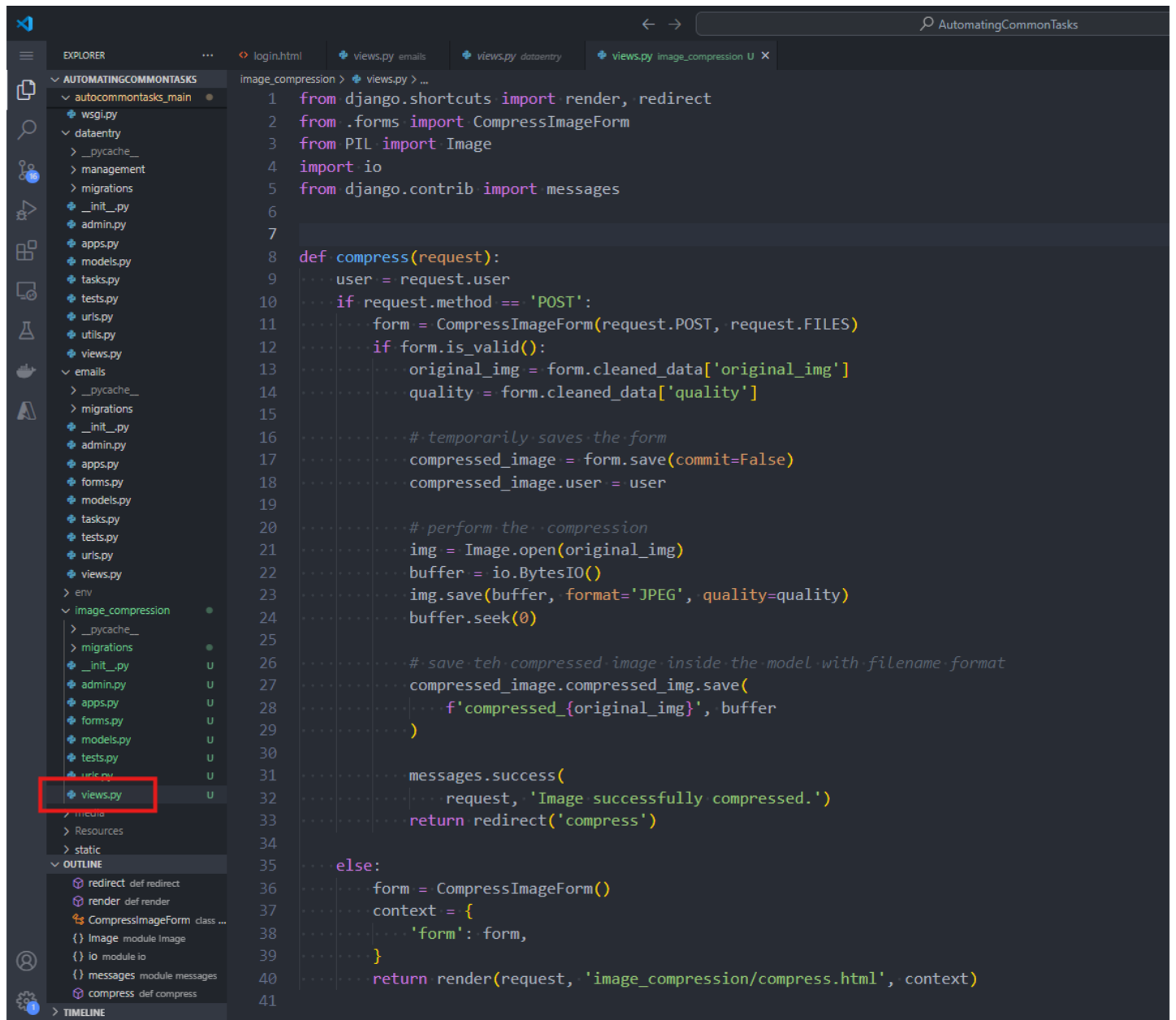
Django version 4.2.14, using settings 'autocommontasks_main.settings'

Starting development server at http://127.0.0.1:8000/

Quit the server with CTRL-BREAK.

```
buffer cursor position or pointer at the beginning = > 0
buffer cursor position or pointer after image compression = > 1226514
buffer cursor position or pointer after return to zero = > 0
Internal Server Error: /image-compression/compress/
Traceback (most recent call last):
File "C:\Users\Rosillie\OneDrive\Desktop\LEARNING DJANGO PROJECTS\AutomatingCommonTasks\env\lib\site-packages\django\db\backends\utils.py", line 89, in _execute
```

17. The VIEWS.PY shall be:



18. Checking our ADMIN panel

← → ↻ ⓘ http://127.0.0.1:8000/admin/image_compression/compressimage/5/change/

Django administration

Home > Image_Compression > Compress images > commontask_admin

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

Change compress image

commontask_admin

User: commontask_admin ✎ + 👁

Original img: Currently: original_images/flat-500-4322521_cpdmALp.jpg

Change: No file chosen

Quality: 30 ▼

Compressed img: Currently: compressed_images/compressed_flat-500-4322521_gFLYuX3.jpg

Change: No file chosen

19. To set the image format to any format not just JPEG, we update our VIEWS.PY AS:

```
... # perform the compression
... img = Image.open(original_img)

... # set the image format based on the uploaded image's format
... output_format = img.format

... buffer = io.BytesIO()
... img.save(buffer, format=output_format, quality=quality)
... buffer.seek(0)

... # save the compressed image inside the model with filename format
```