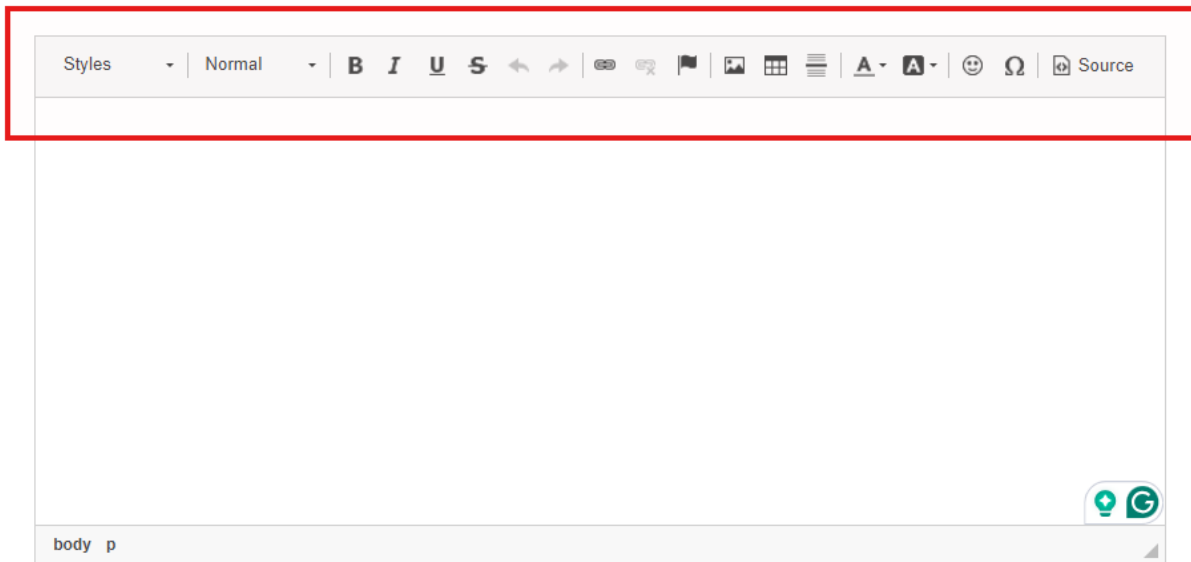


Topic: Bulk Email Tool 19: Integrating Rich Text Editor

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



The email's body should have a formatting toolbar just like the image below and we can use CKEDITOR with this. This is the same tool we used in our [Django Blog](#) and this [Digital Notebook](#) website.



1. See this [CKEditor documentation](#) for the details of installation and configuration.

Required

1. Install or add django-ckeditor to your python path.

```
pip install django-ckeditor
```

2. Add `ckeditor` to your `INSTALLED_APPS` setting.
3. Run the `collectstatic` management command: `$./manage.py collectstatic`. This will copy static CKEditor required media resources into the directory given by the `STATIC_ROOT` setting. See [Django's documentation on managing static files](#) for more info.

Install CkEditor in your Django-bash terminal

```
$ pip install django-ckeditor
```

2. Add CkEditor in the INSTALLED APPS in SETTINGS.PY

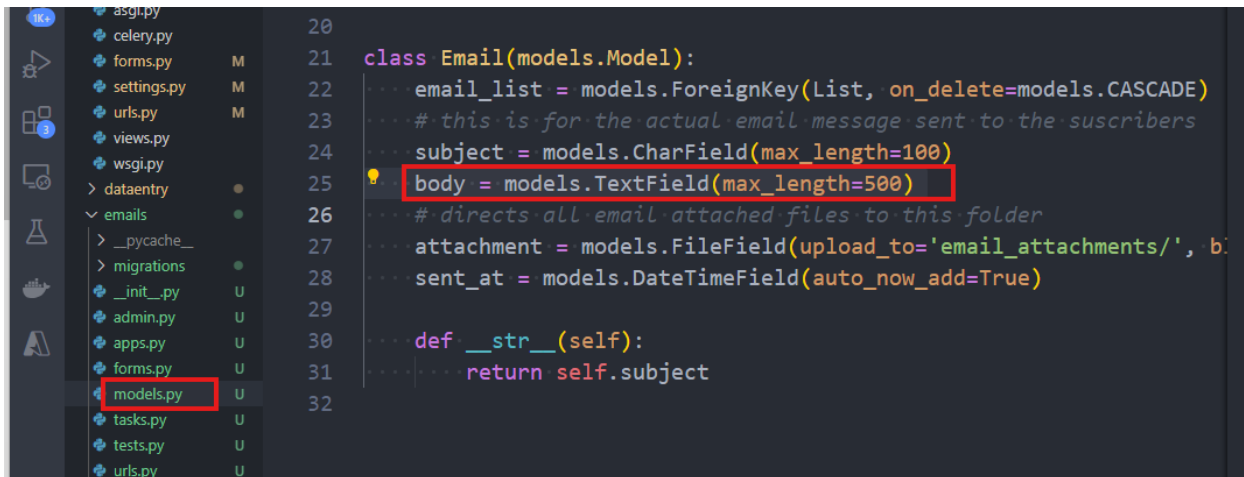
```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'dataentry',
    'uploads',
    'crispy_forms',
    'crispy_bootstrap5',
    'emails',
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

3. Run the COLLECTSTATIC COMMAND.

```
$ python manage.py collectstatic
```

4. Now update the EMAIL model in EMAILS\MODELS.PY, that will require Rick Text Editor:

FROM:



```
20
21 class Email(models.Model):
22     email_list = models.ForeignKey(List, on_delete=models.CASCADE)
23     # this is for the actual email message sent to the subscribers
24     subject = models.CharField(max_length=100)
25     body = models.TextField(max_length=500)
26     # directs all email attached files to this folder
27     attachment = models.FileField(upload_to='email_attachments/', b:
28     sent_at = models.DateTimeField(auto_now_add=True)
29
30     def __str__(self):
31         return self.subject
32
```

TO:

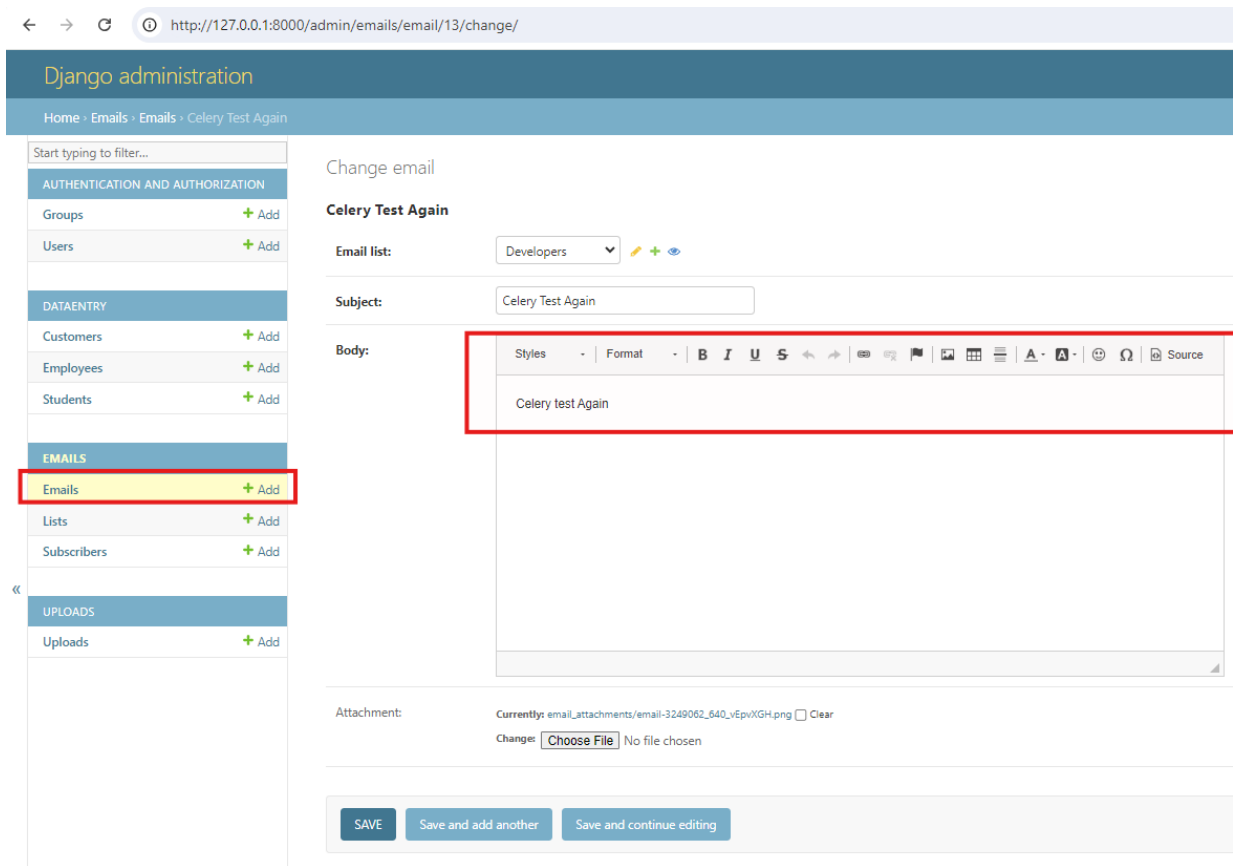
```
AutomatingCommonTasks
EXPLORER login.html views.py U tasks.py U settings.py M utils.py M models.py U X send-
AUTOMATINGCOMMONT... emails > models.py > Email > body
1 from django.db import models
2 from ckeditor.fields import RichTextField
3
4
5 class List(models.Model):
6     # this is the email list
7     email_list = models.CharField(max_length=25)
8
9     def __str__(self):
10        return self.email_list
11
12
13 class Subscriber(models.Model):
14     # this email address can be part of any List
15     email_list = models.ForeignKey(List, on_delete=models.CASCADE)
16     email_address = models.EmailField(max_length=50)
17
18     def __str__(self):
19        return self.email_address
20
21
22 class Email(models.Model):
23     email_list = models.ForeignKey(List, on_delete=models.CASCADE)
24     # this is for the actual email message sent to the subscribers
25     subject = models.CharField(max_length=100)
26     body = RichTextField()
27     # directs all email attached files to this folder
28     attachment = models.FileField(upload_to='email_attachments/', b
29     sent_at = models.DateTimeField(auto_now_add=True)
30
```

5. Since we have made some changes in our model, run the migration commands.

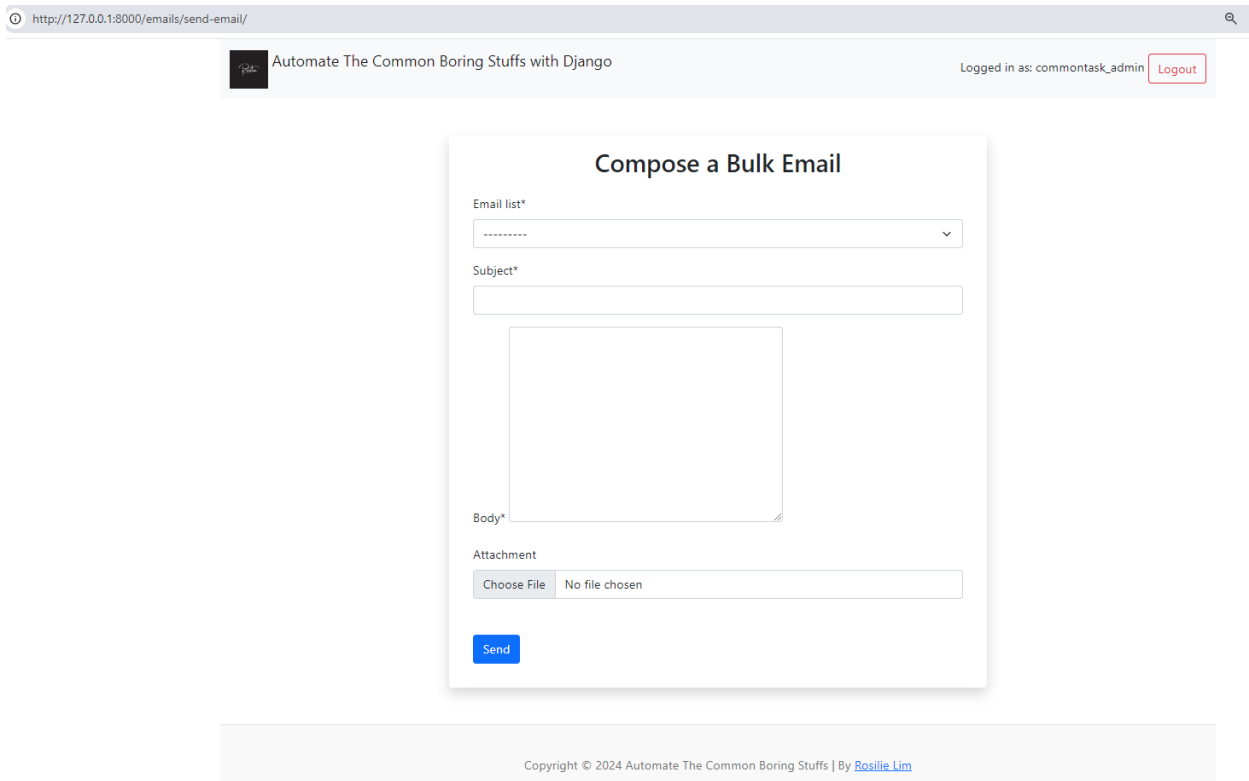
```
$ python manage.py makemigrations
```

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

6. Run your Django server and see your admin dashboard Email model to see the effect of CKEditor.



7. Now we want the CKEditor to load as well on our front-end HTML page. When we load the SEND-EMAIL page, this is now the after-result:



8. In the CKEditor, copy the JAVASCRIPTS LINKS and save this in BASE.HTML

```
<script type="text/javascript" src="{% static "ckeditor/ckeditor-init.js" %}"></script>
```

```
<script type="text/javascript" src="{% static "ckeditor/ckeditor/ckeditor.js" %}"></script>
```

Outside of django admin

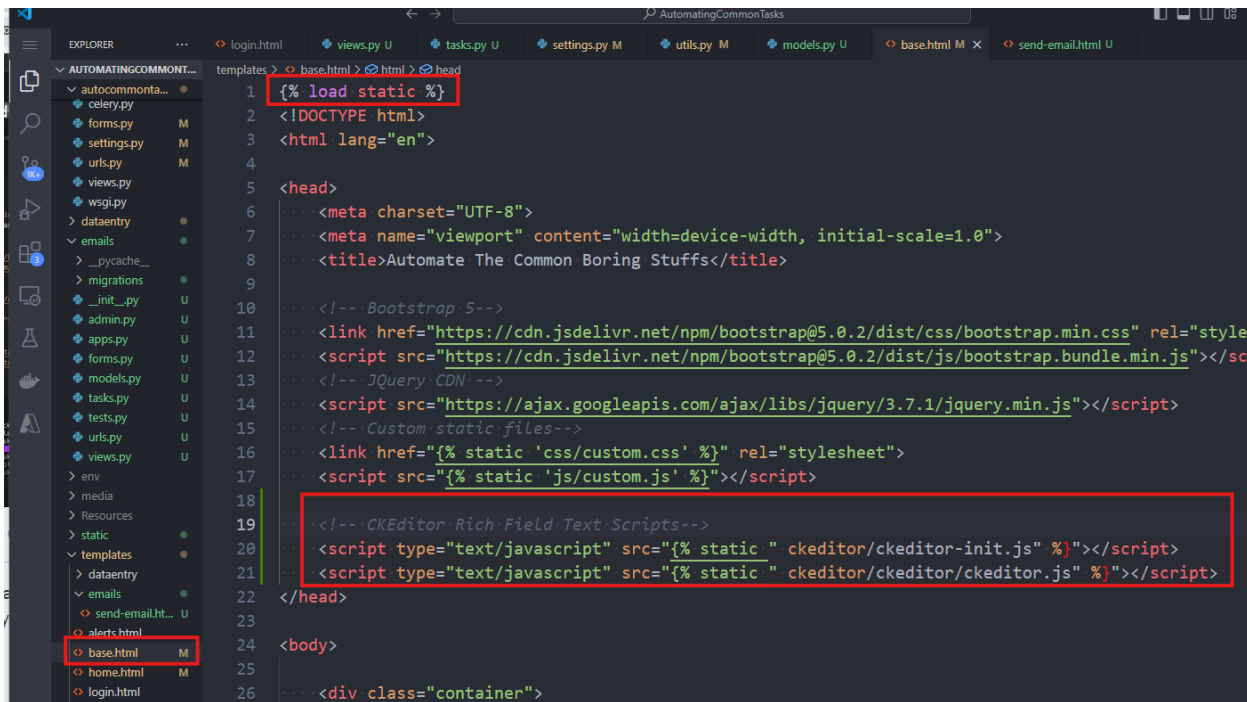
When you are rendering a form outside the admin panel, you'll have to make sure all form media is present for the editor to work. One way to achieve this is like this:

```
<form>
  {{ myform.media }}
  {{ myform.as_p }}
  <input type="submit"/>
</form>
```

or you can load the media manually as it is done in the demo app:

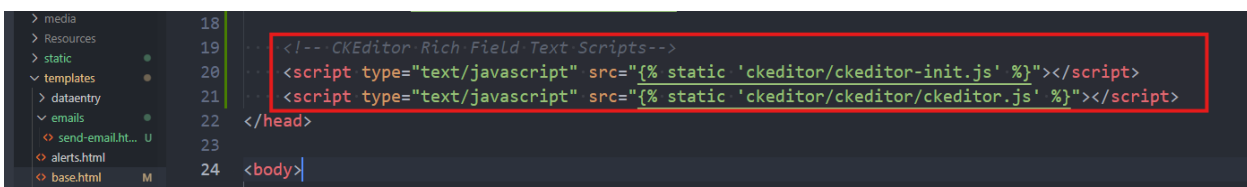
```
{% load static %}
<script type="text/javascript" src="{% static "ckeditor/ckeditor-init.js" %}"></script>
<script type="text/javascript" src="{% static "ckeditor/ckeditor/ckeditor.js" %}"></script>
```

When you need to render `RichTextField`'s HTML output in your templates safely, just use `{{ content|safe }}`, [Django's safe filter](#)



```
1  {% load static %}
2  <!DOCTYPE html>
3  <html lang="en">
4
5  <head>
6    <!-- Bootstrap 5-->
7    <meta charset="UTF-8">
8    <meta name="viewport" content="width=device-width, initial-scale=1.0">
9    <title>Automate The Common Boring Stuffs</title>
10   <!-- Custom static files-->
11   <link href="{% static 'css/custom.css' %}" rel="stylesheet">
12   <script src="{% static 'js/custom.js' %}"></script>
13   <!-- CKEditor Rich Field Text Scripts-->
14   <script type="text/javascript" src="{% static "ckeditor/ckeditor-init.js" %}"></script>
15   <script type="text/javascript" src="{% static "ckeditor/ckeditor/ckeditor.js" %}"></script>
16 </head>
17
18 <body>
19   <div class="container">
```

The CKEditor scripts are showing red-highlighted symbols or words. This is because of the double " " used in the line. So modify it as:



```
18 <!-- CKEditor Rich Field Text Scripts-->
19 <script type="text/javascript" src="{% static 'ckeditor/ckeditor-init.js' %}"></script>
20 <script type="text/javascript" src="{% static 'ckeditor/ckeditor/ckeditor.js' %}"></script>
21 </head>
22
23 <body>
```

9. Now refresh our SEND-EMAIL page and the Rich Field Toolbar should show up:



Compose a Bulk Email

Email list*

Subject*

Body*

Styles - | Format - | **B** *I* U ~~S~~ ← → | 📧 📎 🏠 | 🖼️ 📄 📑 | A · A · |

😊 Ω | 📄 Source

Attachment
 No file chosen

10. The CKEditor has a default size window which can be resizeable but if you want a different height, we can configure it as:

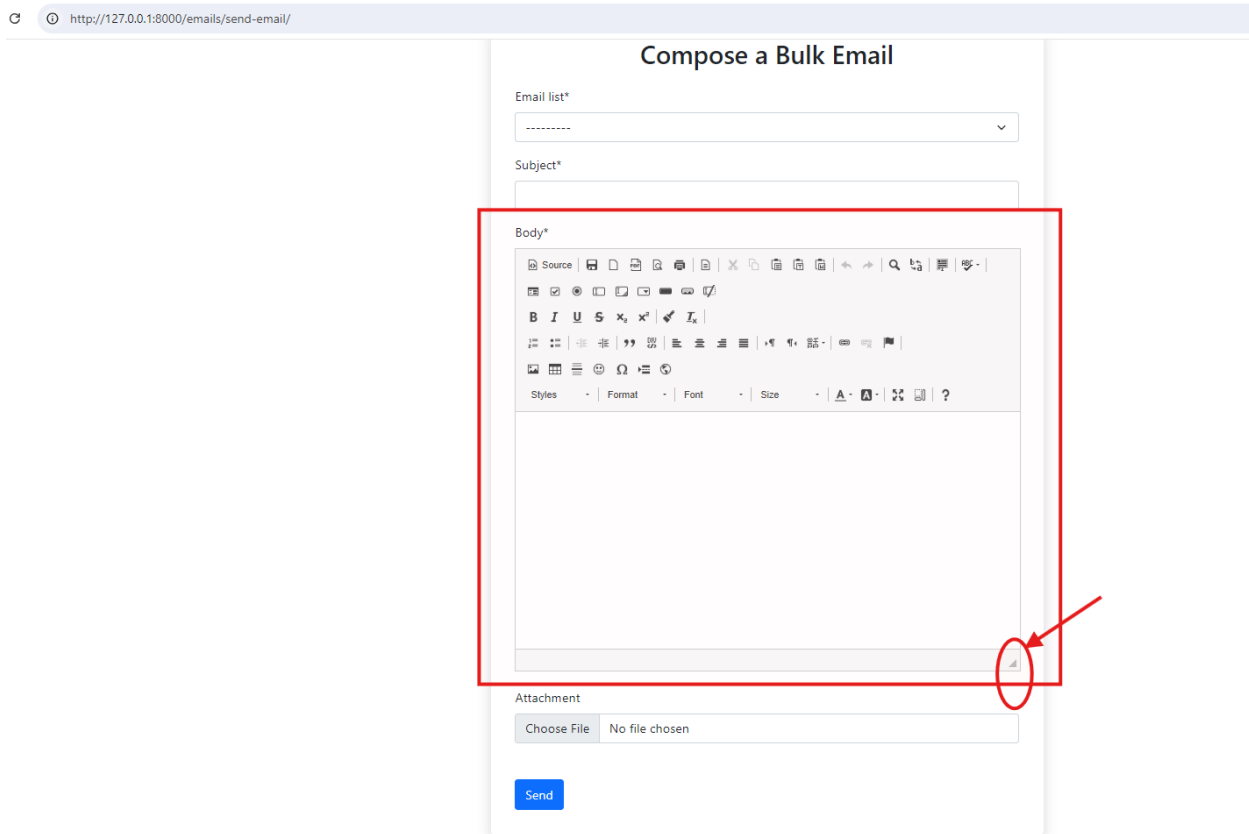
By specifying a set named `default` you'll be applying its settings to all `RichTextField` and `CKEditorWidget` objects for which `config_name` has not been explicitly defined

```
CKEDITOR_CONFIGS = {
    'default': {
        'toolbar': 'full',
        'height': 300,
        'width': 300,
    },
}
```

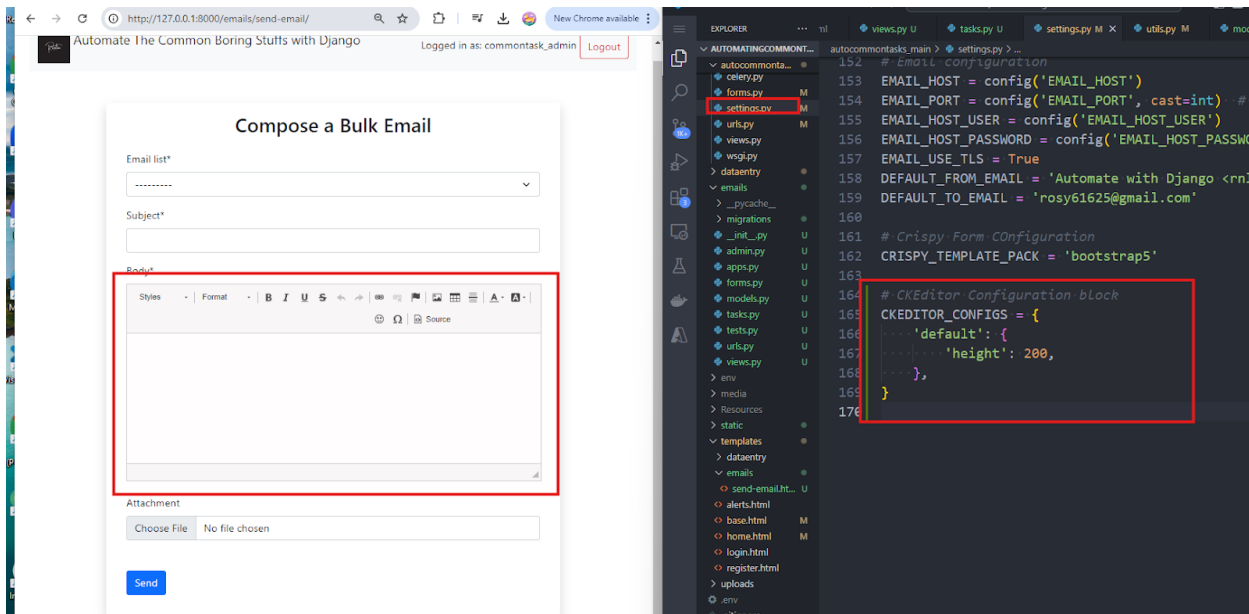
So, go to `SETTINGS.PY` and add the above configuration block.

```
152 # Email configuration
153 EMAIL_HOST = config('EMAIL_HOST')
154 EMAIL_PORT = config('EMAIL_PORT', cast=int) # converts to int
155 EMAIL_HOST_USER = config('EMAIL_HOST_USER')
156 EMAIL_HOST_PASSWORD = config('EMAIL_HOST_PASSWORD')
157 EMAIL_USE_TLS = True
158 DEFAULT_FROM_EMAIL = 'Automate with Django <rnldesolutions@gmail.com>'
159 DEFAULT_TO_EMAIL = 'rosy61625@gmail.com'
160
161 # Crispy Form Configuration
162 CRISPY_TEMPLATE_PACK = 'bootstrap5'
163
164 # CKEditor Configuration block
165 CKEDITOR_CONFIGS = {
166     'default': {
167         'toolbar': 'full',
168         'height': 300,
169     },
170 }
171
```

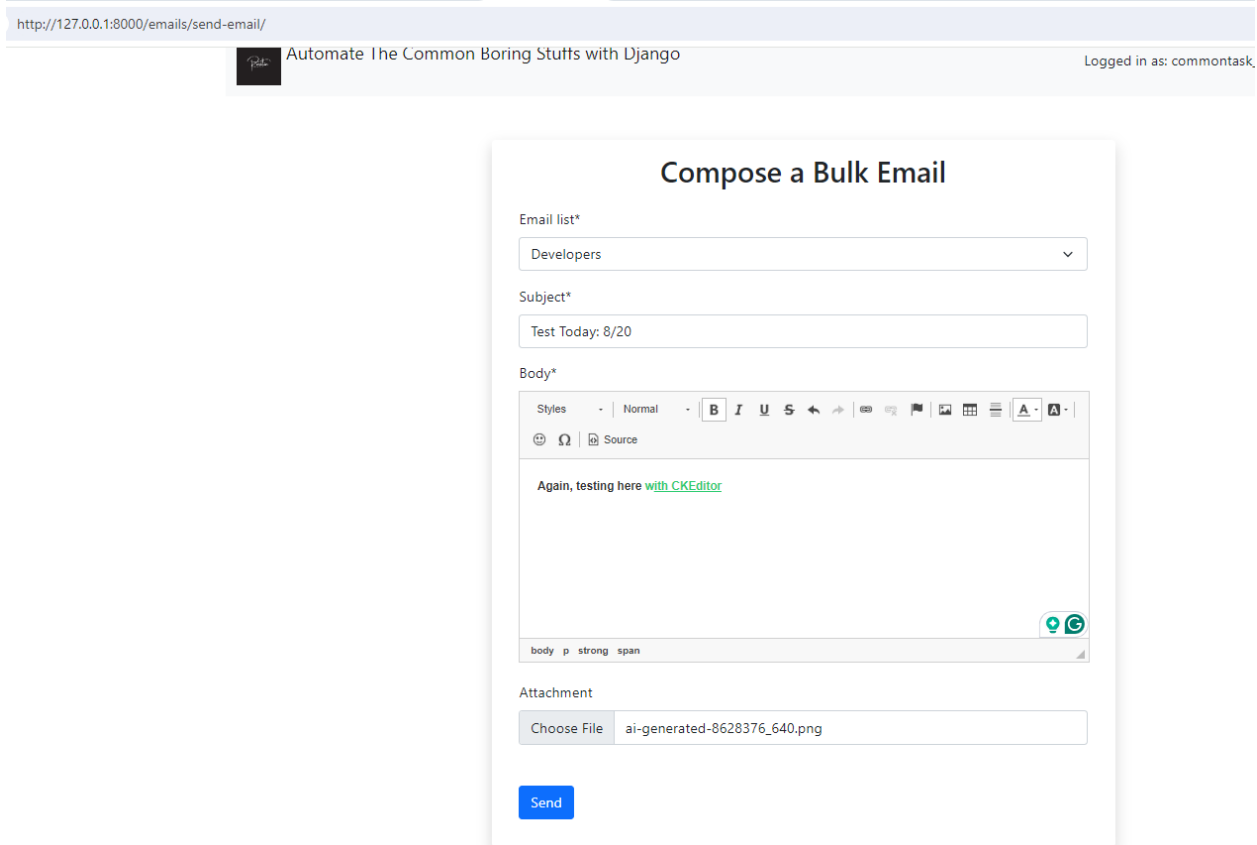
Reload the page and you can see the complete tools with a height. Take note that we can resize the window using the bottom right corner.



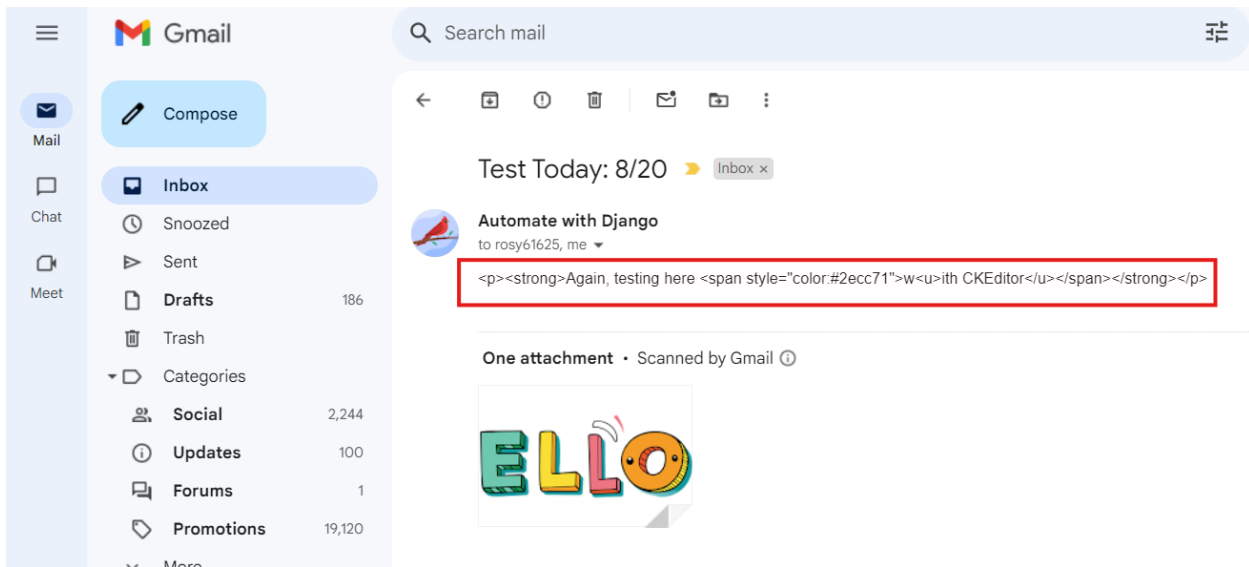
11. Showing a complete toolbar would be too much, so we simplify it by using the default and with reduced height.



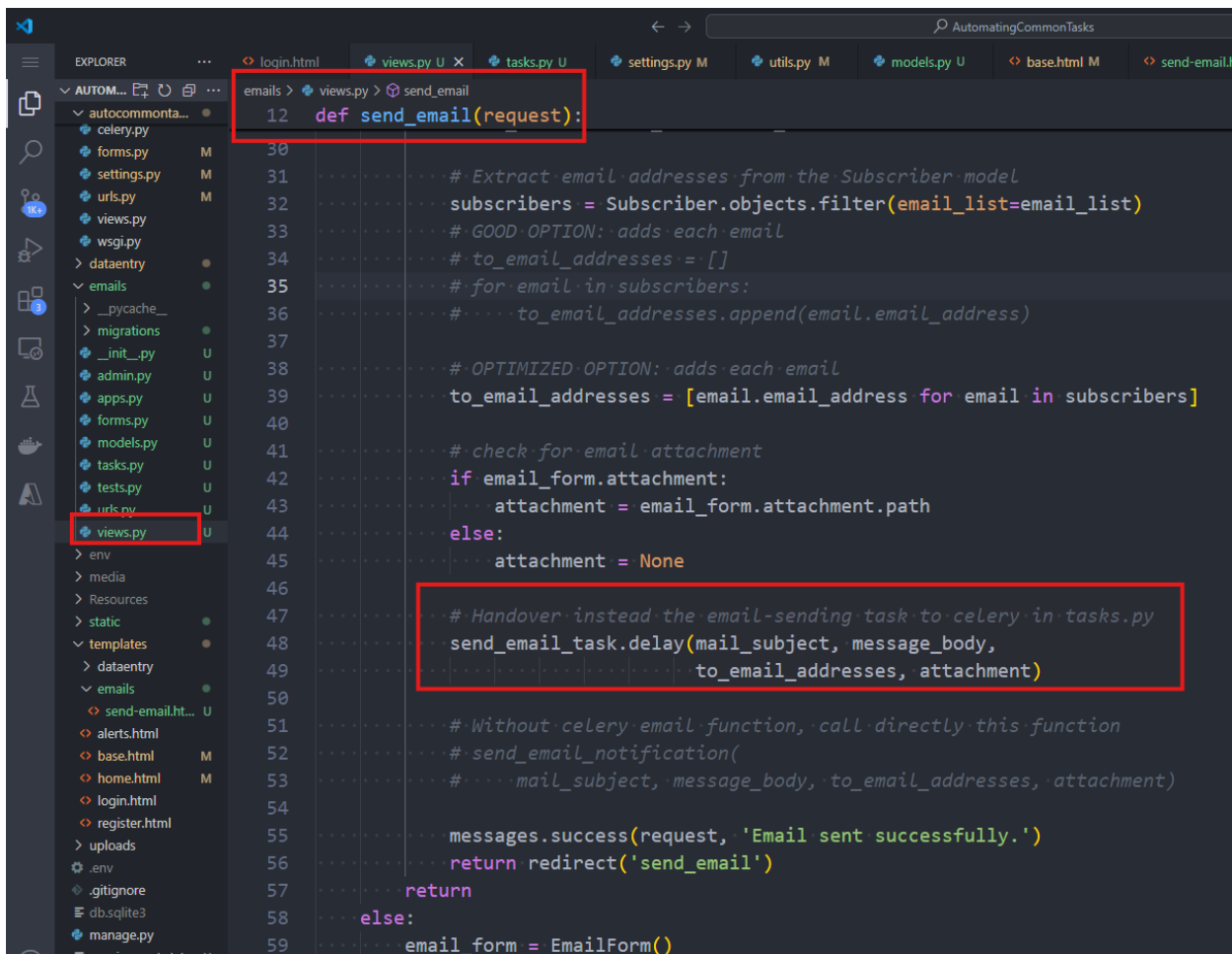
12. However, when we send the emails, we will be receiving it with HTML tags.



Checking our email:



13. Go to your SEND_EMAIL function in EMAILSVIEWS.PY and press CTRL + click the link on the SEND_EMAIL_TASK to bring you to the function.



In EMAILSTASKS.PY, press CTRL + LEFT CLICK on our SEND_EMAIL_NOTIFICATION function to bring us to another function:

```

1 # from the main project folder within CELERY.PY, we load the variable APP
2 from autocommtasks_main.celery import app
3 from dataentry.utils import send_email_notification
4
5
6 @app.task
7 def send_email_task(mail_subject, message_body, to_email_addresses, attachment):
8     send_email_notification(mail_subject, message_body,
9                             to_email_addresses, attachment)
10
11     return 'Email sending task executed successfully.'

```

In DATAENTRYUTILS.PY, we update our function to show that the email body is sent as HTML without the HTML tags.

```

30 def check_csv_error(file_path, model_name):
31     csv_header = reader.fieldnames
32     # compare CSV header with the model's field names and throw appropriate message
33     if csv_header != model_fields:
34         raise DataError(
35             f'CSV file does not match with the {model_name} table fields')
36     except Exception as e:
37         raise e
38     # if there is no error, we return the model name
39     return target_model
40
41 # sends an email with dynamic subject, message and recipient's address
42 # Attachment as an optional value
43
44 def send_email_notification(mail_subject, message_body, to_email_addresses, attachment=None):
45     try:
46         # calls our environment variable, DEFAULT_FROM_EMAIL from settings.py
47         from_email = settings.DEFAULT_FROM_EMAIL
48         mail = EmailMessage(mail_subject, message_body,
49                             from_email, to=to_email_addresses)
50         # if file is present as part of our parameters.
51         if attachment is not None:
52             mail.attach_file(attachment)
53
54         # send the mail as HTML content without the HTML tags
55         mail.content_subtype = "html"
56
57         mail.send() # sends our mail
58     except Exception as e:

```

14. We have to reload our CELERY bash terminal code since we have made some changes in our Celery function. Press CTRL + C to stop it from running:

```

67
68
69 def send_email_notification(mail_subject, message_body, to_email_addresses, attachment=None):
70     try:
71         # calls our environment variable, DEFAULT_FROM_EMAIL from settings.py
72         from_email = settings.DEFAULT_FROM_EMAIL
73         mail = EmailMessage(mail_subject, message_body,
74                             from_email, to=to_email_addresses)
75
76         # send the mail as HTML content without the HTML tags
77         mail.content_subtype = "html"
78
79         mail.send() # sends our mail
80     except Exception as e:

```

```

[2024-08-18 22:03:10,796: INFO/MainProcess] Task emails.tasks.send_email_task[e4e64a91-a7e1-4c77-a0e9-e8e4dae0fd11] received
[2024-08-18 22:03:13,132: INFO/MainProcess] Task emails.tasks.send_email_task[e4e64a91-a7e1-4c77-a0e9-e8e4dae0fd11] succeeded in 2.34299999999934807s: 'Email sending task executed successfully.'
[2024-08-20 13:12:14,476: INFO/MainProcess] Task emails.tasks.send_email_task[b989a302-b04a-4f7d-be32-86e23099e5d] received
[2024-08-20 13:12:14,476: INFO/MainProcess] Task emails.tasks.send_email_task[b989a302-b04a-4f7d-be32-86e23099e5d] succeeded in 2.34299999999934807s: 'Email sending task executed successfully.'

worker: Hitting ctrl+c again will terminate all running tasks!
worker: Warm shutdown (MainProcess)
(env)
$ celery -A autocommtasks main worker --loglevel=info --pool=solo

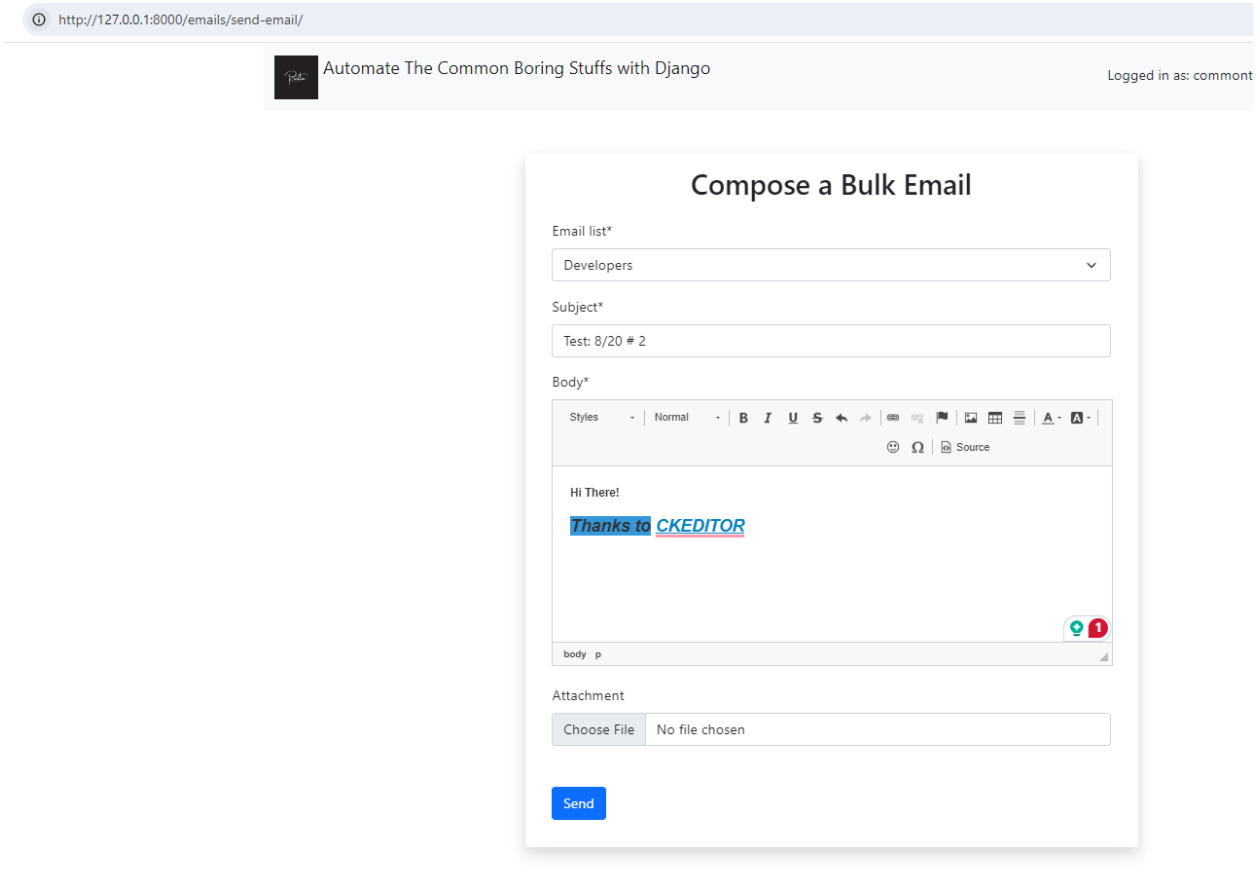
```

We encountered an error: CKEditor MODULE NOT FOUND Error, so we installed it as well:

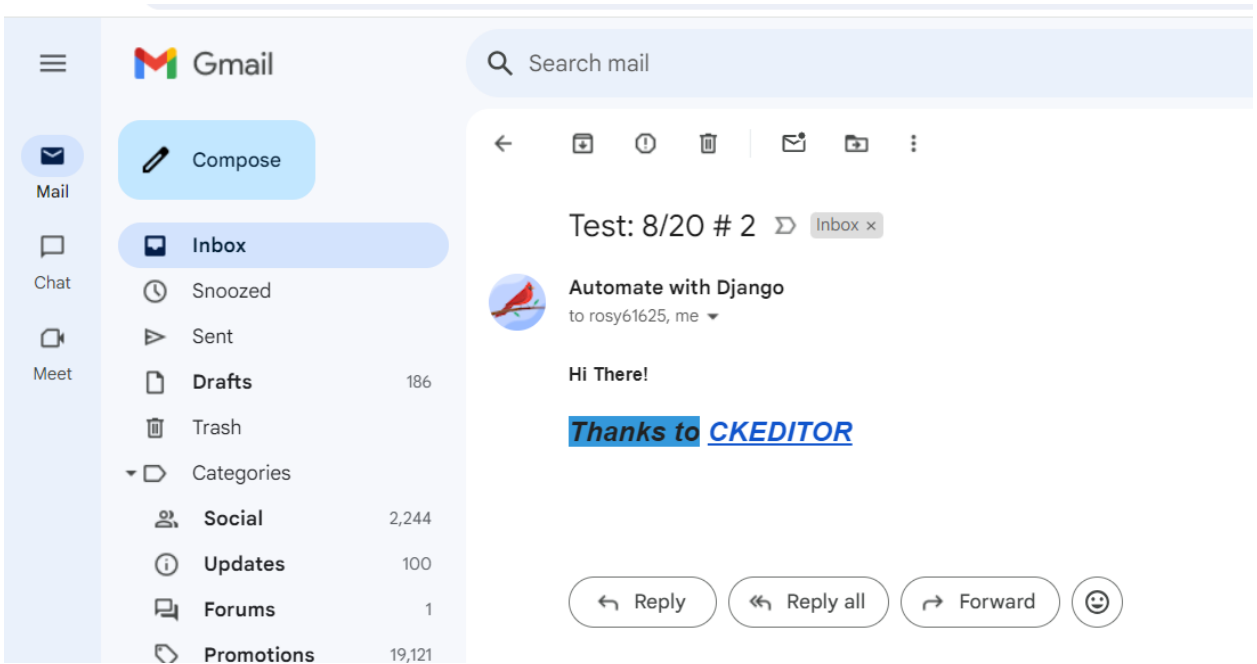
\$ pip install django-ckeditor

```
File "C:\Users\Rosllie\AppData\Local\Programs\Python\Python39\lib\importlib\_init_.py", line 127, in import_module
    return _bootstrap._gcd_import(name[level:], package, level)
File "<frozen importlib._bootstrap>", line 1039, in _gcd_import
File "<frozen importlib._bootstrap>", line 1007, in _find_and_load
File "<frozen importlib._bootstrap>", line 984, in _find_and_load_unlocked
ModuleNotFoundError: No module named 'ckeditor'
(env)
Rosllie@DELL MINGW64 C:/Users/Rosllie/AppData/Local/Programs/Microsoft VS Code (main)
$ pip install django-ckeditor
Collecting django-ckeditor
  Using cached django_ckeditor-6.7.1-py3-none-any.whl.metadata (32 kB)
Requirement already satisfied: Django>=3.2 in c:\users\rosllie\appdata\local\programs\python\python39\lib\site-packages (from django-ckeditor) (4.2.14)
Collecting django-is-asset>=2.0 (from django-ckeditor)
  Collecting django-is-asset>=2.0 (from django-ckeditor)
```

15. Sending our email again,



In our email:



16. Push our changes in Github.

