**Integrate CKEditor 5 with Django 5 Rich Text Editor**

**(atlast I give me the simple ckeditor method )**

Video 46

Code 39

When first we canupload dat through ckeditor and than we want to access it on our web page than django provide the one tag

{{form.content|safe}}

SEARCH CK EDITOR IN PYPI.ORG ON INTERNET

pip install django-ckeditor-5==0.2.15

and

pip install pillow==11.0.0

**complete method with ck editor 5**

and than make file in project :

ckeditorconfig.py:

from urllib.parse import urljoin

import os

from django.conf import settings

from django.core.files.storage import FileSystemStorage

customColorPalette = [

        {

            'color': 'hsl(4, 90%, 58%)',

            'label': 'Red'

        },

        {

            'color': 'hsl(340, 82%, 52%)',

            'label': 'Pink'

        },

        {

            'color': 'hsl(291, 64%, 42%)',

            'label': 'Purple'

        },

        {

            'color': 'hsl(262, 52%, 47%)',

            'label': 'Deep Purple'

        },

        {

            'color': 'hsl(231, 48%, 48%)',

            'label': 'Indigo'

        },

        {

            'color': 'hsl(207, 90%, 54%)',

            'label': 'Blue'

        },

    ]

CKEDITOR\_5\_CONFIGS = {

  'default': {

      'toolbar': ['heading', '|', 'bold', 'italic', 'link',

                  'bulletedList', 'numberedList', 'blockQuote', 'imageUpload', ],

  },

  'extends': {

      'blockToolbar': [

          'paragraph', 'heading1', 'heading2', 'heading3',

          '|',

          'bulletedList', 'numberedList',

          '|',

          'blockQuote',

      ],

      'toolbar': ['heading', '|', 'outdent', 'indent', '|', 'bold', 'italic', 'link', 'underline', 'strikethrough',

      'code','subscript', 'superscript', 'highlight', '|', 'codeBlock', 'sourceEditing', 'insertImage',

                  'bulletedList', 'numberedList', 'todoList', '|',  'blockQuote', 'imageUpload', '|',

                  'fontSize', 'fontFamily', 'fontColor', 'fontBackgroundColor', 'mediaEmbed', 'removeFormat',

                  'insertTable',],

      'image': {

          'toolbar': ['imageTextAlternative', '|', 'imageStyle:alignLeft',

                      'imageStyle:alignRight', 'imageStyle:alignCenter', 'imageStyle:side',  '|'],

          'styles': [

              'full',

              'side',

              'alignLeft',

              'alignRight',

              'alignCenter',

          ]

      },

      'table': {

          'contentToolbar': [ 'tableColumn', 'tableRow', 'mergeTableCells',

          'tableProperties', 'tableCellProperties' ],

          'tableProperties': {

              'borderColors': customColorPalette,

              'backgroundColors': customColorPalette

          },

          'tableCellProperties': {

              'borderColors': customColorPalette,

              'backgroundColors': customColorPalette

          }

      },

      'heading' : {

          'options': [

              { 'model': 'paragraph', 'title': 'Paragraph', 'class': 'ck-heading\_paragraph' },

              { 'model': 'heading1', 'view': 'h1', 'title': 'Heading 1', 'class': 'ck-heading\_heading1' },

              { 'model': 'heading2', 'view': 'h2', 'title': 'Heading 2', 'class': 'ck-heading\_heading2' },

              { 'model': 'heading3', 'view': 'h3', 'title': 'Heading 3', 'class': 'ck-heading\_heading3' }

          ]

      }

  },

  'list': {

      'properties': {

          'styles': 'true',

          'startIndex': 'true',

          'reversed': 'true',

      }

  }

}

# Define a constant in settings.py to specify file upload permissions

CKEDITOR\_5\_FILE\_UPLOAD\_PERMISSION = "staff"  # Possible values: "staff", "authenticated", "any"

CKEDITOR\_5\_CUSTOM\_CSS = 'blog/css/ckeditor.css' # optional

# Define the path where CKEditor uploads should go

CKEDITOR\_UPLOAD\_PATH = "post\_image/"

# Define the custom storage class to use this path

class PostImageStorage(FileSystemStorage):

    def \_\_init\_\_(self, \*args, \*\*kwargs):

        kwargs['location'] = os.path.join(

            settings.MEDIA\_ROOT, CKEDITOR\_UPLOAD\_PATH)

        kwargs['base\_url'] = urljoin(

            settings.MEDIA\_URL, CKEDITOR\_UPLOAD\_PATH)

        super().\_\_init\_\_(\*args, \*\*kwargs)

CKEDITOR\_5\_FILE\_STORAGE = "ch38.ckeditorconfig.PostImageStorage" # optional

And in setting.py:

"""

Django settings for ch39 project.

Generated by 'django-admin startproject' using Django 5.1.3.

For more information on this file, see

https://docs.djangoproject.com/en/5.1/topics/settings/

For the full list of settings and their values, see

https://docs.djangoproject.com/en/5.1/ref/settings/

"""

from pathlib import Path

from .ckeditorconfig import \*

# Build paths inside the project like this: BASE\_DIR / 'subdir'.

BASE\_DIR = Path(\_\_file\_\_).resolve().parent.parent

# Quick-start development settings - unsuitable for production

# See https://docs.djangoproject.com/en/5.1/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!

SECRET\_KEY = 'django-insecure-ovj^+f38z)@xtp3exrnpyjc2wp5y2qkl\*!)75bpwnve4bu!k\_f'

# SECURITY WARNING: don't run with debug turned on in production!

DEBUG = True

ALLOWED\_HOSTS = []

# Application definition

INSTALLED\_APPS = [

    'django.contrib.admin',

    'django.contrib.auth',

    'django.contrib.contenttypes',

    'django.contrib.sessions',

    'django.contrib.messages',

    'django.contrib.staticfiles',

    'django\_ckeditor\_5',  # here we add ckeditor 5

    'blog',

]

MIDDLEWARE = [

    'django.middleware.security.SecurityMiddleware',

    'django.contrib.sessions.middleware.SessionMiddleware',

    'django.middleware.common.CommonMiddleware',

    'django.middleware.csrf.CsrfViewMiddleware',

    'django.contrib.auth.middleware.AuthenticationMiddleware',

    'django.contrib.messages.middleware.MessageMiddleware',

    'django.middleware.clickjacking.XFrameOptionsMiddleware',

]

ROOT\_URLCONF = 'ch39.urls'

TEMPLATES = [

    {

        'BACKEND': 'django.template.backends.django.DjangoTemplates',

        'DIRS': [],

        'APP\_DIRS': True,

        'OPTIONS': {

            'context\_processors': [

                'django.template.context\_processors.debug',

                'django.template.context\_processors.request',

                'django.contrib.auth.context\_processors.auth',

                'django.contrib.messages.context\_processors.messages',

            ],

        },

    },

]

WSGI\_APPLICATION = 'ch38.wsgi.application'

# Database

# https://docs.djangoproject.com/en/5.1/ref/settings/#databases

DATABASES = {

    'default': {

        'ENGINE': 'django.db.backends.sqlite3',

        'NAME': BASE\_DIR / 'db.sqlite3',

    }

}

# Password validation

# https://docs.djangoproject.com/en/5.1/ref/settings/#auth-password-validators

AUTH\_PASSWORD\_VALIDATORS = [

    {

        'NAME': 'django.contrib.auth.password\_validation.UserAttributeSimilarityValidator',

    },

    {

        'NAME': 'django.contrib.auth.password\_validation.MinimumLengthValidator',

    },

    {

        'NAME': 'django.contrib.auth.password\_validation.CommonPasswordValidator',

    },

    {

        'NAME': 'django.contrib.auth.password\_validation.NumericPasswordValidator',

    },

]

# Internationalization

# https://docs.djangoproject.com/en/5.1/topics/i18n/

LANGUAGE\_CODE = 'en-us'

TIME\_ZONE = 'UTC'

USE\_I18N = True

USE\_TZ = True

# Static files (CSS, JavaScript, Images)

# https://docs.djangoproject.com/en/5.1/howto/static-files/

STATIC\_URL = 'static/'

MEDIA\_URL = '/media/'

MEDIA\_ROOT = BASE\_DIR / 'media'

# Default primary key field type

# https://docs.djangoproject.com/en/5.1/ref/settings/#default-auto-field

DEFAULT\_AUTO\_FIELD = 'django.db.models.BigAutoField'

Urls.py:

from django.contrib import admin

from django.urls import path, include

from django.conf import settings

from django.conf.urls.static import static

urlpatterns = [

    path('admin/', admin.site.urls),

    path("ckeditor5/", include('django\_ckeditor\_5.urls')),

    path('', include('blog.urls')),

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

App/Models.py:

from django.db import models

from django\_ckeditor\_5.fields import CKEditor5Field

class Post(models.Model):

  title = models.CharField(max\_length=255)

  content = CKEditor5Field(config\_name='extends')

App/forms.py:

from django import forms

from blog.models import Post

from django\_ckeditor\_5.widgets import CKEditor5Widget

class CreatePostForm(forms.ModelForm):

  def \_\_init\_\_(self, \*args, \*\*kwargs):

    super().\_\_init\_\_(\*args, \*\*kwargs)

    self.fields["content"].required = False

  class Meta:

    model = Post

    fields = ['title', 'content']

    # widgets = {

    #               "content": CKEditor5Widget(

    #                   attrs={"class": "django\_ckeditor\_5"}, config\_name="extends"

    #               )

    #           }

    content = CKEditor5Widget(attrs={"class": "django\_ckeditor\_5"}, config\_name="extends")

app /views.py:

from django.shortcuts import render, redirect

from blog.forms import CreatePostForm

from blog.models import Post

def home(request):

    if request.method == "POST":

        form = CreatePostForm(request.POST)  # no files, since CKEditor5 uses HTML content

        if form.is\_valid():

            form.save()  # saves the Post object

            return redirect('home')  # redirect to same page after saving

    else:

        form = CreatePostForm()

    # Get all blog posts to display below the form

    blog\_posts = Post.objects.all()

    return render(request, 'blog/home.html', {'form': form, 'blog\_posts': blog\_posts})

in this in our super user admin in dark mode much error in text so the solution is here make css file in

static->app->css->ckeditor.css

ckeditor.css:

@media (prefers-color-scheme: dark) {

  .ck.ck-editor {

      color: black;

  }

  /\* This makes editor scrollable \*/

  .ck-editor\_\_editable {

    min-height: 100px;

    max-height: 100px;

}

/\* This applies padding in editor so list bullet and number will else it hides on screen\*/

.ck-editor\_\_editable\_inline{

  padding: 0 30px !important;

}

  :root {

      /\* Overrides the border radius setting in the theme. \*/

      --ck-border-radius: 4px;

      /\* Overrides the default font size in the theme. \*/

      --ck-font-size-base: 14px;

      /\* Helper variables to avoid duplication in the colors. \*/

      --ck-custom-background: hsl(270, 1%, 29%);

      --ck-custom-foreground: hsl(255, 3%, 18%);

      --ck-custom-border: hsl(300, 1%, 22%);

      --ck-custom-white: hsl(0, 0%, 100%);

      /\* -- Overrides generic colors. ------------------------------------------------------------- \*/

      --ck-color-base-foreground: var(--ck-custom-background);

      --ck-color-focus-border: hsl(208, 90%, 62%);

      --ck-color-text: hsl(0, 0%, 98%);

      --ck-color-shadow-drop: hsla(0, 0%, 0%, 0.2);

      --ck-color-shadow-inner: hsla(0, 0%, 0%, 0.1);

      /\* -- Overrides the default .ck-button class colors. ---------------------------------------- \*/

      --ck-color-button-default-background: var(--ck-custom-background);

      --ck-color-button-default-hover-background: hsl(270, 1%, 22%);

      --ck-color-button-default-active-background: hsl(270, 2%, 20%);

      --ck-color-button-default-active-shadow: hsl(270, 2%, 23%);

      --ck-color-button-default-disabled-background: var(--ck-custom-background);

      --ck-color-button-on-background: var(--ck-custom-foreground);

      --ck-color-button-on-hover-background: hsl(255, 4%, 16%);

      --ck-color-button-on-active-background: hsl(255, 4%, 14%);

      --ck-color-button-on-active-shadow: hsl(240, 3%, 19%);

      --ck-color-button-on-disabled-background: var(--ck-custom-foreground);

      --ck-color-button-action-background: hsl(168, 76%, 42%);

      --ck-color-button-action-hover-background: hsl(168, 76%, 38%);

      --ck-color-button-action-active-background: hsl(168, 76%, 36%);

      --ck-color-button-action-active-shadow: hsl(168, 75%, 34%);

      --ck-color-button-action-disabled-background: hsl(168, 76%, 42%);

      --ck-color-button-action-text: var(--ck-custom-white);

      --ck-color-button-save: hsl(120, 100%, 46%);

      --ck-color-button-cancel: hsl(15, 100%, 56%);

      /\* -- Overrides the default .ck-dropdown class colors. -------------------------------------- \*/

      --ck-color-dropdown-panel-background: var(--ck-custom-background);

      --ck-color-dropdown-panel-border: var(--ck-custom-foreground);

      /\* -- Overrides the default .ck-splitbutton class colors. ----------------------------------- \*/

      --ck-color-split-button-hover-background: var(--ck-color-button-default-hover-background);

      --ck-color-split-button-hover-border: var(--ck-custom-foreground);

      /\* -- Overrides the default .ck-input class colors. ----------------------------------------- \*/

      --ck-color-input-background: var(--ck-custom-background);

      --ck-color-input-border: hsl(257, 3%, 43%);

      --ck-color-input-text: hsl(0, 0%, 98%);

      --ck-color-input-disabled-background: hsl(255, 4%, 21%);

      --ck-color-input-disabled-border: hsl(250, 3%, 38%);

      --ck-color-input-disabled-text: hsl(0, 0%, 78%);

      /\* -- Overrides the default .ck-labeled-field-view class colors. ---------------------------- \*/

      --ck-color-labeled-field-label-background: var(--ck-custom-background);

      /\* -- Overrides the default .ck-list class colors. ------------------------------------------ \*/

      --ck-color-list-background: var(--ck-custom-background);

      --ck-color-list-button-hover-background: var(--ck-color-base-foreground);

      --ck-color-list-button-on-background: var(--ck-color-base-active);

      --ck-color-list-button-on-background-focus: var(--ck-color-base-active-focus);

      --ck-color-list-button-on-text: var(--ck-color-base-background);

      /\* -- Overrides the default .ck-balloon-panel class colors. --------------------------------- \*/

      --ck-color-panel-background: var(--ck-custom-background);

      --ck-color-panel-border: var(--ck-custom-border);

      /\* -- Overrides the default .ck-toolbar class colors. --------------------------------------- \*/

      --ck-color-toolbar-background: var(--ck-custom-background);

      --ck-color-toolbar-border: var(--ck-custom-border);

      /\* -- Overrides the default .ck-tooltip class colors. --------------------------------------- \*/

      --ck-color-tooltip-background: hsl(252, 7%, 14%);

      --ck-color-tooltip-text: hsl(0, 0%, 93%);

      /\* -- Overrides the default colors used by the ckeditor5-image package. --------------------- \*/

      --ck-color-image-caption-background: hsl(0, 0%, 97%);

      --ck-color-image-caption-text: hsl(0, 0%, 20%);

      /\* -- Overrides the default colors used by the ckeditor5-widget package. -------------------- \*/

      --ck-color-widget-blurred-border: hsl(0, 0%, 87%);

      --ck-color-widget-hover-border: hsl(43, 100%, 68%);

      --ck-color-widget-editable-focus-background: var(--ck-custom-white);

      /\* -- Overrides the default colors used by the ckeditor5-link package. ---------------------- \*/

      --ck-color-link-default: hsl(190, 100%, 75%);

  }

}

And than come to

ckeditorconfig.py(and add this code)

CKEDITOR\_5\_CUSTOM\_CSS = 'blog/css/ckeditor.css'

And if we want to make foler name media and than store things in so also add in that folder :

For production level we can make changgs in this folder so we can chagethings in this folder during proudction like set this in aws storage or other s

# Define the path where CKEditor uploads should go

CKEDITOR\_UPLOAD\_PATH = "post\_image/"

# Define the custom storage class to use this path

class PostImageStorage(FileSystemStorage):

    def \_\_init\_\_(self, \*args, \*\*kwargs):

        kwargs['location'] = os.path.join(

            settings.MEDIA\_ROOT, CKEDITOR\_UPLOAD\_PATH)

        kwargs['base\_url'] = urljoin(

            settings.MEDIA\_URL, CKEDITOR\_UPLOAD\_PATH)

        super().\_\_init\_\_(\*args, \*\*kwargs)

upper in complete code we also write these things

and also update this folder at the end file name

CKEDITOR\_5\_FILE\_STORAGE = "ch39.ckeditorconfig.PostImageStorage"

# ✅ Integrate CKEditor OLD with Django 5

### 1️⃣ Install CKEditor

pip install django-ckeditor

### 2️⃣ settings.py

(only add CKEditor + Media + Static related parts)

INSTALLED\_APPS = [

'ckeditor',

'ckeditor\_uploader',

'blog', # your app

]

# Static files

STATIC\_URL = '/static/'

STATICFILES\_DIRS = [BASE\_DIR / "static"]

STATIC\_ROOT = BASE\_DIR / "staticfiles"

# Media files

MEDIA\_URL = '/media/'

MEDIA\_ROOT = BASE\_DIR / "media"

# CKEditor settings

CKEDITOR\_UPLOAD\_PATH = "uploads/"

CKEDITOR\_CONFIGS = {

'default': {

'toolbar': 'full',

'height': 300,

'width': '100%',

},

}

### 3️⃣ project/urls.py

(add ckeditor + media serving)

from django.contrib import admin

from django.urls import path, include

from django.conf import settings

from django.conf.urls.static import static

urlpatterns = [

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

path('', include('blog.urls')),

path('ckeditor/', include('ckeditor\_uploader.urls')),

]

if settings.DEBUG:

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

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

### 4️⃣ app/models.py

from django.db import models

from ckeditor\_uploader.fields import RichTextUploadingField

class Post(models.Model):

title = models.CharField(max\_length=255)

content = RichTextUploadingField() # ✅ allows images/files upload

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

return self.title

### 5️⃣ app/forms.py

from django import forms

from blog.models import Post

from ckeditor\_uploader.widgets import CKEditorUploadingWidget

class CreatePostForm(forms.ModelForm):

content = forms.CharField(widget=CKEditorUploadingWidget()) # ✅ use uploader widget

class Meta:

model = Post

fields = ['title', 'content']

### 6️⃣ app/views.py

from django.shortcuts import render, redirect

from blog.forms import CreatePostForm

from blog.models import Post

def home(request):

if request.method == "POST":

form = CreatePostForm(request.POST, request.FILES)

if form.is\_valid():

form.save()

return redirect('home')

else:

form = CreatePostForm()

posts = Post.objects.all()

return render(request, "blog/home.html", {"form": form, "posts": posts})

### 7️⃣ app/urls.py

from blog.views import home

from django.urls import path

urlpatterns = [

path('', home, name='home'),

]

### 8️⃣ template (blog/home.html)

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Blog Post</title>

{{ form.media }} <!-- ✅ very important for CKEditor -->

</head>

<body>

<h1>Create Blog Post:</h1>

<form method="POST" enctype="multipart/form-data">

{% csrf\_token %}

{{ form.as\_p }}

<button type="submit">Save</button>

</form>

<hr>

<h2>All Posts</h2>

{% for post in posts %}

<h3>{{ post.title }}</h3>

<div>{{ post.content|safe }}</div> <!-- ✅ safe to render HTML -->

{% endfor %}

</body>

</html>

### 9️⃣ Final Steps

python manage.py collectstatic

python manage.py makemigrations

python manage.py migrate