django-quill-editor

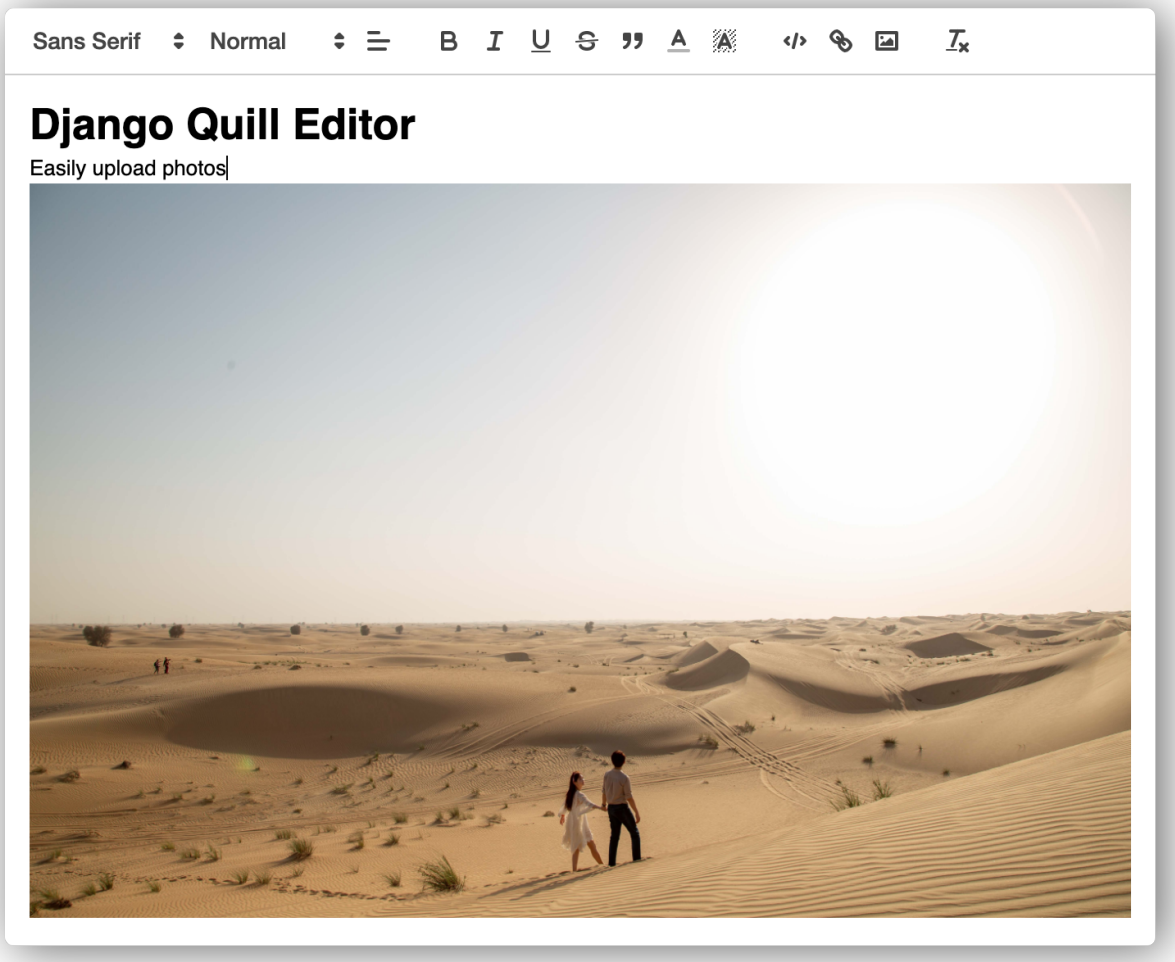
Release 0.1

Feb 07, 2023

Contents

django-quill-editor makes *Quill.js* easy to use on Django Forms and admin sites

- **No configuration required for static files!**
- The entire code for inserting WYSIWYG editor is less than 30 lines
- It can be used in both admin and Django views



CHAPTER 1

Using in Django admin

Add `QuillField` to the **Model class** you want to use

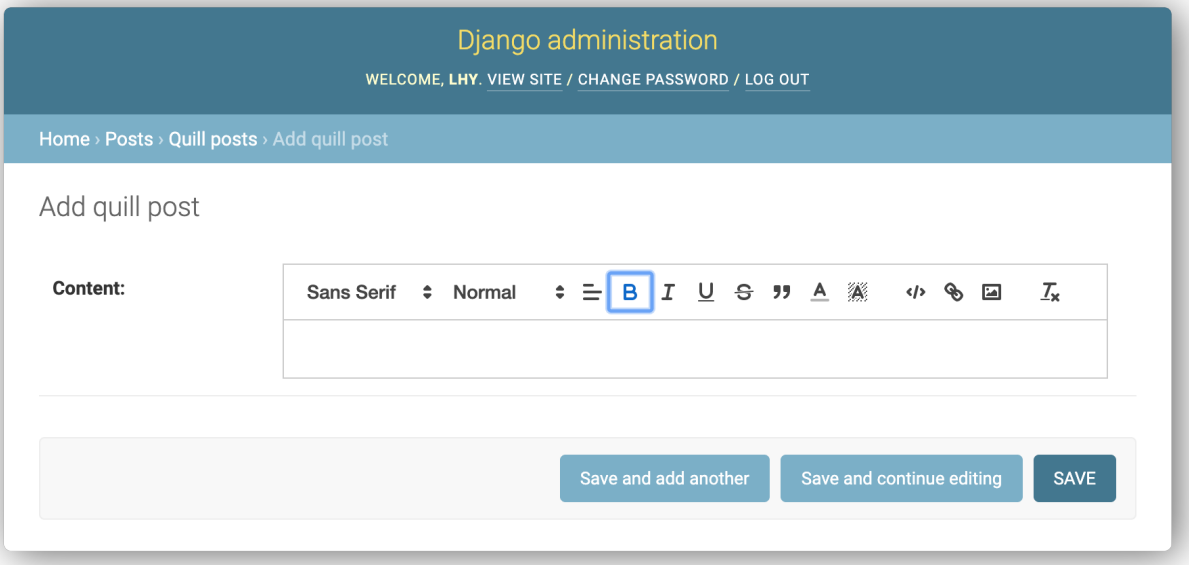
```
# models.py
from django.db import models
from django_quill.fields import QuillField

class QuillPost(models.Model):
    content = QuillField()
```

Just register the Model in **admin.py** of the app

```
from django.contrib import admin
from .models import QuillPost

@admin.register(QuillPost)
class QuillPostAdmin(admin.ModelAdmin):
    pass
```



admin-sample

CHAPTER 2

Using as Form

Add the CSS and JS to the `<head>` of the template or base template.

There are two ways to add CSS and JS files to a template.

- If there is a **Form** with QuillField added, add `{{ form.media }}` to the `<head>` tag.

```
<head>
    {{ form.media }}
</head>
```

- Or, import CSS and JS files directly using `{% include %}` template tags.

```
<head>
    <!-- django-quill-editor Media -->
    {% include 'django_quill/media.html' %}
</head>
```

Add `QuillFormField` to the **Form class** you want to use.

```
# forms.py
from django import forms
from django_quill.forms import QuillFormField

class QuillFieldForm(forms.Form):
    content = QuillFormField()
```

Add a **Form instance** containing **QuillFormField** to the template context in the view.

```
# views.py
from django.shortcuts import render
from .forms import QuillFieldForm

def form_view(request):
    return render(request, 'form_view.html', {'form': QuillFieldForm()})
```

In the template, use the received **Form instance** variable. (in the above case, **'form'**)

```
<!-- form_view.html -->
<form action="" method="POST">{% csrf_token %}
    {{ form.content }}
</form>
```

CHAPTER 3

Using as ModelForm

Add `QuillField` to the **Model class** you want to use

```
# models.py
from django.db import models
from django_quill.fields import QuillField

class QuillPost(models.Model):
    content = QuillField()
```

Just define and use **ModelForm** of Model class (with QuillField)

```
# forms.py
from django import forms
from .models import QuillPost

class QuillPostForm(forms.ModelForm):
    class Meta:
        model = QuillPost
        fields = (
            'content',
        )
```

Set the **view** and **template** in the same way as when using a **normal Form**

```
# views.py
from django.shortcuts import render
from .forms import QuillPostForm

def model_form_view(request):
    return render(request, 'form_view.html', {'form': QuillPostForm()})
```

```
<!-- form_view.html -->
<form action="" method="POST">{% csrf_token %}
```

(continues on next page)

(continued from previous page)

```
    {{ form.content }}
</form>
```

CHAPTER 4

Change toolbar config

More settings can be found on the official sitehttps://quilljs.com/docs/modules/toolbar/

Add `QUILL_CONFIGS` to the **settings.py**

```
QUILL_CONFIGS = {
    'default':{
        'theme': 'snow',
        'modules': {
            'syntax': True,
            'toolbar': [
                [
                    {'font': []},
                    {'header': []},
                    {'align': []},
                    'bold', 'italic', 'underline', 'strike', 'blockquote',
                    {'color': []},
                    {'background': []},
                ],
                ['code-block', 'link'],
                ['clean'],
            ]
        }
    }
}
```

CHAPTER 5

Migrating to QuillField

To convert a field used as a **TextField** or **CharField** to a **QuillField**, the following operations are required.

Suppose you had a model that looked like this.

```
# Assuming this is the model you created in the posts app
class NonQuillPost(models.Model):
    content = models.TextField()
```

1. Convert existing data into a format suitable for QuillField using django command.

   USE WITH CAUTION This command permanently transforms the contents of that field in the DB.

   ```
   # Usage: python manage.py convert_to_quill {app_name} {model_name} {field_name}
   > python manage.py convert_to_quill posts NonQuillPost content
   ```

2. Change existing field to QuillField.

   ```
   from django_quill.fields import QuillField

   class NonQuillPost(models.Model):
       content = QuillField()
   ```

3. Create a migration.

   ```
   > python manage.py makemigrations
   Migrations for 'posts':
     posts/migrations/0002_alter_nonquillpost_content.py
       - Alter field content on nonquillpost
   ```

4. Apply the migration.

   ```
   > python manage.py migrate
   Operations to perform:
     Apply all migrations: admin, auth, contenttypes, posts, sessions
   ```

(continues on next page)

(continued from previous page)

```
Running migrations:
  Applying posts.0002_alter_nonquillpost_content... OK
```

CHAPTER 6

Installation

Use pip to install from PyPI:

```
pip install django-quill-editor
```

Add **django_quill** to **INSTALLED_APPS** in **settings.py**:

```
INSTALLED_APPS = [
    'django.contrib.admin',
    ...
    'django_quill',
]
```

CHAPTER 7

Contributing

To contribute to **django-quill-editor** create a fork on GitHub. Clone your fork, make some changes, and submit a pull request.

CHAPTER 8

Issues

Use the GitHub issue tracker for **django-quill-editor** to submit bugs, issues, and feature requests.

CHAPTER 9

Indices and tables

- genindex
- modindex
- search