Taming rich text content in Django

Django Day Copenhagen 2023

Let’s talk about prose

What is prose?

- It’s a strange word.

- It is written or spoken language in its ordinary form, without metrical structure.

An example of written prose in rich text

We built Django Prose in order to provide us with a clean, robust and secure full stack solution to manage rich text content in our Django projects. Until now, we found no existing solution that covered all these needs. Let's run through them quickly one by one.

Django Prose is super clean. All you need to do is:

Install it. Example: poetry add django-prose.
Add it to Django's INSTALLED_APPS settings.
Use the Document model or RichTextField directly in your own models.

For a web application, prose is just multiline text

- We know how to handle multiline text in web applications.
- Plain text is a bit bland though.
- Not all text is cut from the same cloth (e.g. plain text, code, rich text).

Let’s spice up our example

An example of written prose as rich text

We built **Django Prose** in order to provide us with a clean, robust and secure full stack solution to manage rich text content in our Django projects. Until now, we found no existing solution that covered all these needs. Let's run through them quickly one by one.

Django Prose is super clean. All you need to do is:

- Install it. Example: `poetry add django-prose`.
- Add it to Django's `INSTALLED_APPS` settings.
- Use the `Document` model or `RichTextField` directly in your own models.

What is rich text?

Rich text is just text enhanced with formats and styling to help content stand out:

- **Strong** (or bold) text
- Unordered and ordered lists
- Images
- Sub$_{\text{scripts}}$

What’s makes rich text special

What makes rich text special

All text is made equal, but some text is more equal than others.

- Contrary to plain text it is stored encoded and then rendered differently.
- More than often than not, it can be in long-form.
- There are multiple formats and interoperability is not guaranteed.

Rich text formats

There are multiple rich text formats, each one with its pros and cons.

- OOXML (Microsoft Word `.docx`)
- Markdown
- HTML

Rich text in web applications

- A straightforward way to handle rich text in web applications is… HTML.
- We should be cautious, when handling rich text in our web applications.
- We should be **extra cautious**, when handling rich text content from users.

Caveats in rich text

Rich text caveats

- Security
- Performance
- Workflow

Security

Security

Since rich text is rendered, we need to take care of a few things, like:

- XSS attacks
- HTML highjacking
- JavaScript global namespace (`window`) polluting

Innocent rich text

```
Hello <strong>Copenhagen</strong>!

Thank you for this opportunity.
```

Not so innocent rich text: XSS

```
Hello <strong>Copenhagen</strong>!

Thank you for this opportunity.

<script src=”https://pkasid.com/bad.js”></script>
```

Not so innocent rich text: HTML hijacking

```
Hello <strong>Copenhagen</strong>!

Thank you for this opportunity.

</body>

<!-- lol! -->
```

Not so innocent rich text: `window` polutting

```
Hello <strong id=”jQuery”>Copenhagen</strong>!

Thank you for this opportunity.
```

Security tips

- Always sanitize rich text content before storing.
- Consider sanitizing text before rendering.
- For some cases, consider rendering in its own browser context.

Performance

Performance

We should pay close attention to performance, when dealing with rich-text content.

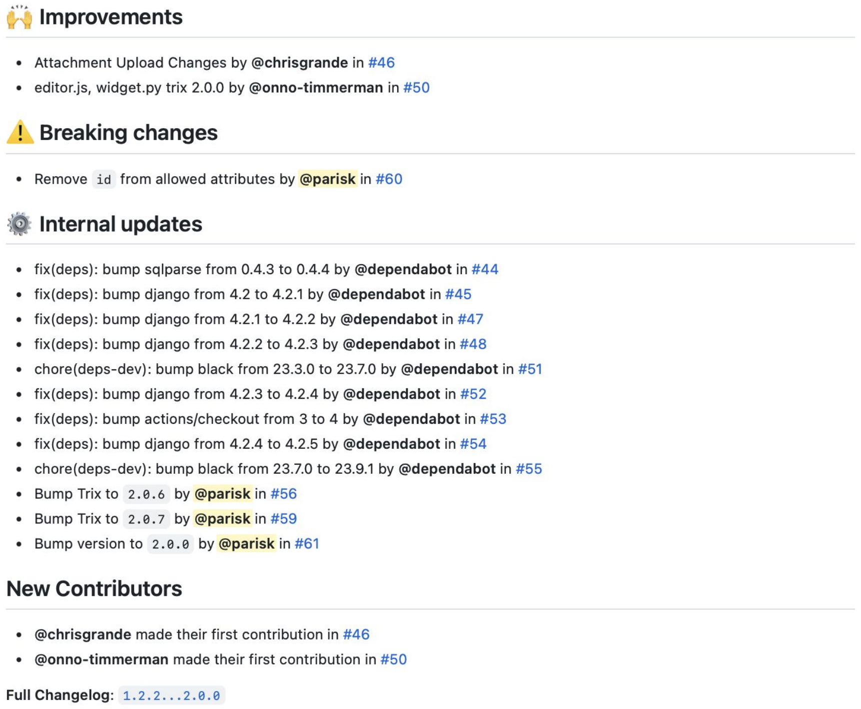
- It takes up more space than its plain text counterpart.
- Its content often is long form.
- More often than not it does not need to be be part of “list queries”.

Performance tips

- Strip it of redundant HTML tags and attributes
- Store it in separate database tables
- Fetch it from the database only when needed

Let’s take an example.

Example GitHub release notes



🙌 Improvements

- Attachment Upload Changes by **@chrisgrande** in #46
- editor.js, widget.py trix 2.0.0 by **@onno-timmerman** in #50

⚠️ Breaking changes

- Remove `id` from allowed attributes by **@parisk** in #60

⚙️ Internal updates

- fix(deps): bump sqlparse from 0.4.3 to 0.4.4 by **@dependabot** in #44
- fix(deps): bump django from 4.2 to 4.2.1 by **@dependabot** in #45
- fix(deps): bump django from 4.2.1 to 4.2.2 by **@dependabot** in #47
- fix(deps): bump django from 4.2.2 to 4.2.3 by **@dependabot** in #48
- chore(deps-dev): bump black from 23.3.0 to 23.7.0 by **@dependabot** in #51
- fix(deps): bump django from 4.2.3 to 4.2.4 by **@dependabot** in #52
- fix(deps): bump actions/checkout from 3 to 4 by **@dependabot** in #53
- fix(deps): bump django from 4.2.4 to 4.2.5 by **@dependabot** in #54
- chore(deps-dev): bump black from 23.7.0 to 23.9.1 by **@dependabot** in #55
- Bump Trix to `2.0.6` by **@parisk** in #56
- Bump Trix to `2.0.7` by **@parisk** in #59
- Bump version to `2.0.0` by **@parisk** in #61

New Contributors

- **@chrisgrande** made their first contribution in #46
- **@onno-timmerman** made their first contribution in #50

Full Changelog: `1.2.2...2.0.0`

1093 bytes of UTF-8 plain text

```
🙌 Improvements

Attachment Upload Changes by @chrisgrande in #46
editor.js, widget.py trix 2.0.0 by @onno-timmerman in #50
⚠️ Breaking changes

Remove id from allowed attributes by @parisk in #60
⚙️ Internal updates

fix(deps): bump sqlparse from 0.4.3 to 0.4.4 by @dependabot in #44
fix(deps): bump django from 4.2 to 4.2.1 by @dependabot in #45
fix(deps): bump django from 4.2.1 to 4.2.2 by @dependabot in #47
fix(deps): bump django from 4.2.2 to 4.2.3 by @dependabot in #48
chore(deps-dev): bump black from 23.3.0 to 23.7.0 by @dependabot in #51
fix(deps): bump django from 4.2.3 to 4.2.4 by @dependabot in #52
fix(deps): bump actions/checkout from 3 to 4 by @dependabot in #53
fix(deps): bump django from 4.2.4 to 4.2.5 by @dependabot in #54
chore(deps-dev): bump black from 23.7.0 to 23.9.1 by @dependabot in #55
Bump Trix to 2.0.6 by @parisk in #56
Bump Trix to 2.0.7 by @parisk in #59
Bump version to 2.0.0 by @parisk in #61
New Contributors

@chrisgrande made their first contribution in #46
@onno-timmerman made their first contribution in #50
Full Changelog: 1.2.2...2.0.0
```

11707 bytes of UTF-8 rich text in HTML format

```
<h2>🙌 Improvements</h2>
<ul>
<li>Attachment Upload Changes by <a class="user-mention notranslate" data-hovercard-type="user"
data-hovercard-url="/users/chrisgrande/hovercard" data-octo-click="hovercard-link-click"
data-octo-dimensions="link_type:self" href="https://github.com/chrisgrande">@chrisgrande</a> in <a
class="issue-link js-issue-link" data-error-text="Failed to load title" data-id="1708161312"
data-permission-text="Title is private" data-url="https://github.com/withlogicco/django-prose/issues/46"
data-hovercard-type="pull_request" data-hovercard-url="/withlogicco/django-prose/pull/46/hovercard"
href="https://github.com/withlogicco/django-prose/pull/46">#46</a></li>
<li>editor.js, widget.py trix 2.0.0 by <a class="user-mention notranslate" data-hovercard-type="user"
data-hovercard-url="/users/onno-timmerman/hovercard" data-octo-click="hovercard-link-click"
data-octo-dimensions="link_type:self" href="https://github.com/onno-timmerman">@onno-timmerman</a> in <a
class="issue-link js-issue-link" data-error-text="Failed to load title" data-id="1789328917"
data-permission-text="Title is private" data-url="https://github.com/withlogicco/django-prose/issues/50"
data-hovercard-type="pull_request" data-hovercard-url="/withlogicco/django-prose/pull/50/hovercard"
href="https://github.com/withlogicco/django-prose/pull/50">#50</a></li>
</ul>

# ...and so on
```

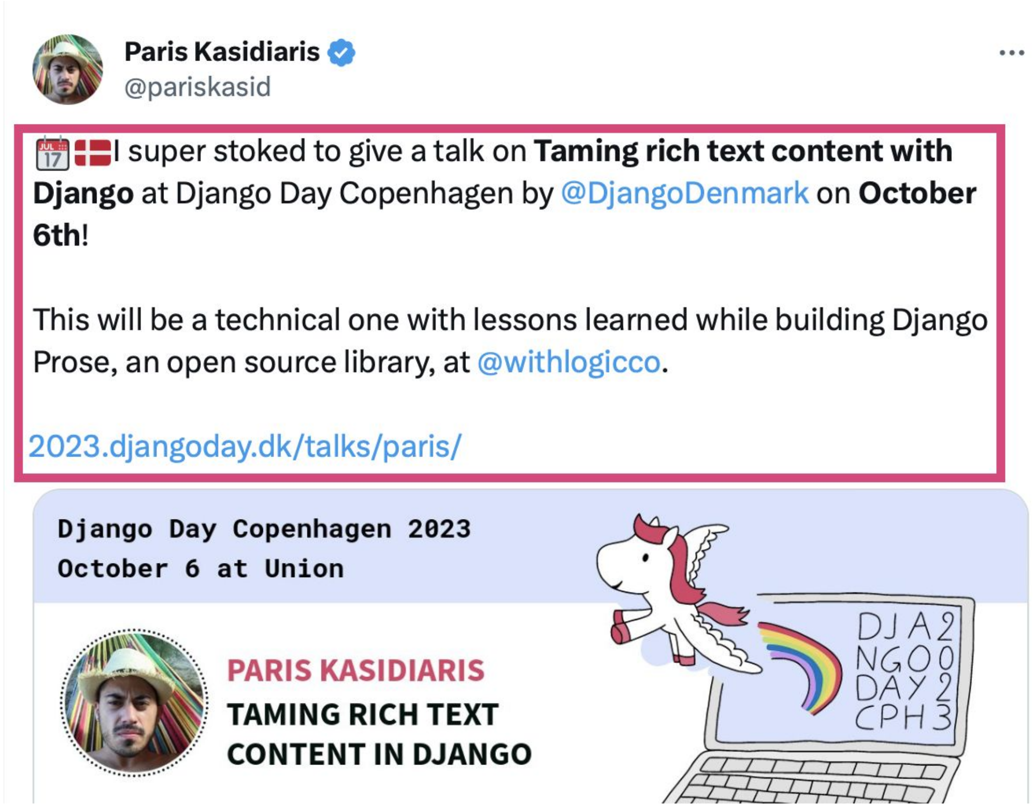
Another example

Another example

Let’s imagine we have an application like X:

- Posts can be up to to ~500 bytes of text.
- Multiple users can post
- Users can reply to posts with posts

This post is about 500 bytes



Paris Kasidiaris
@pariskasid

📆🇩🇰I super stoked to give a talk on **Taming rich text content with Django** at Django Day Copenhagen by @DjangoDenmark on **October 6th**!

This will be a technical one with lessons learned while building Django Prose, an open source library, at @withlogicco.

2023.djangoday.dk/talks/paris/

Django Day Copenhagen 2023
October 6 at Union
PARIS KASIDIARIS
TAMING RICH TEXT CONTENT IN DJANGO
DJANGO DAY CPH 2023

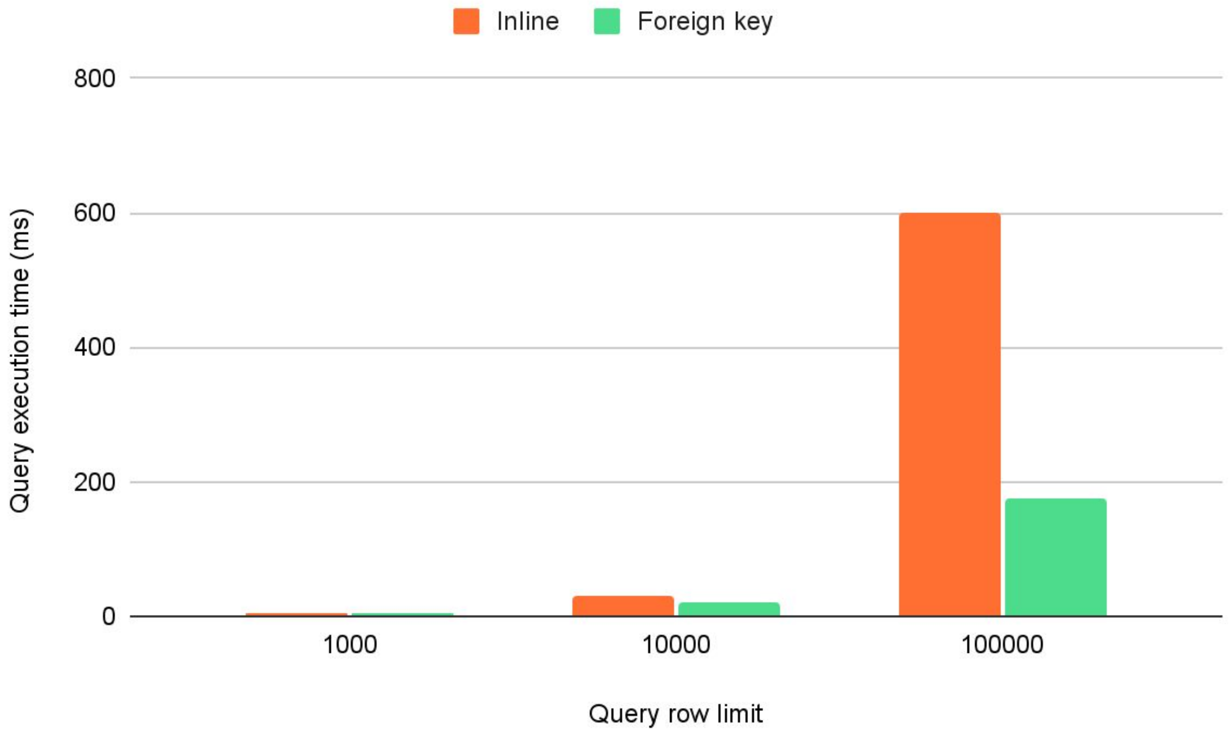
Performance tests

We will compare the following

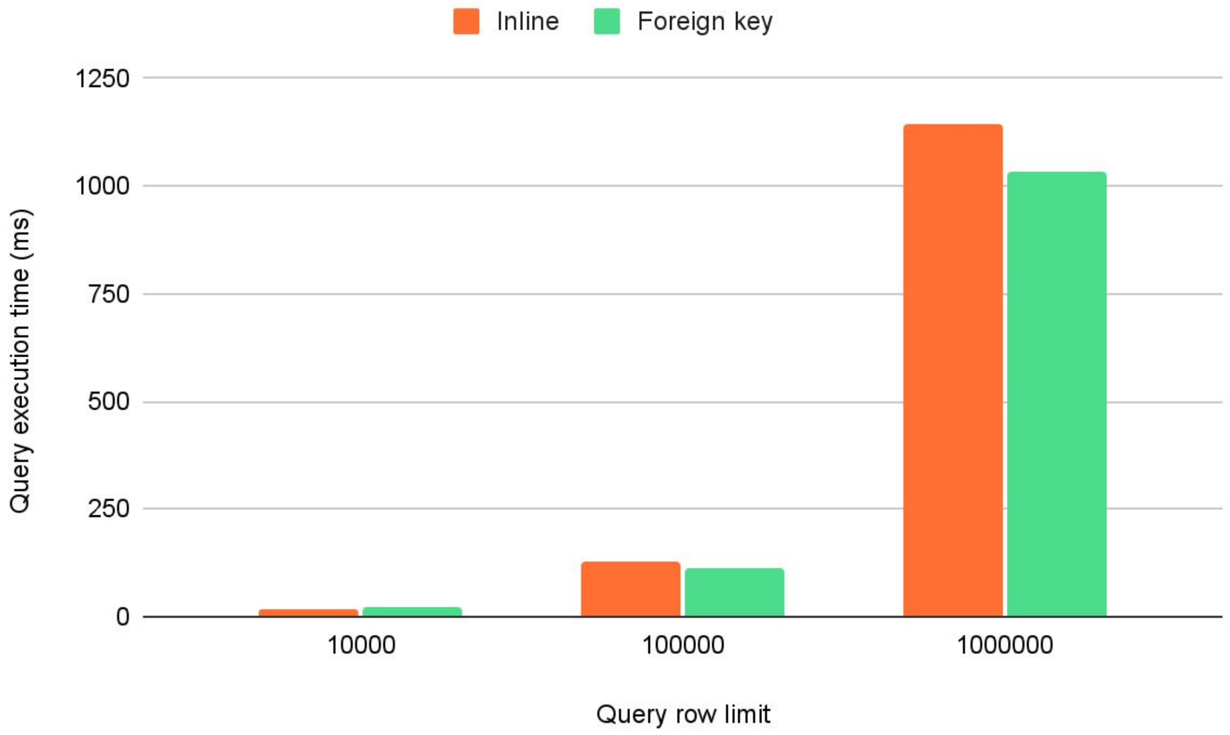
- Two different tables; inline rich-text content and foreign-key external
- Two different queries: `select *` and `select id, title`

A picture is a thousand words

SELECT * FROM table



`SELECT id, title FROM table`



Workflow

Workflow

All of these need to be taken care of by our development workflow.

- User friendly experience with an appropriate editor.
- Crystal clear data format. We should know what is safe and what is not.
- Transparent management of security and performance, where possible.

It’s about time we talk Django

Django Prose

We built an open source library that helped us achieve the above and is

- Easy to integrate and use
- Secure
- Robust

What’s in the box

- Field and model for storing rich text content.
- Widget for editing rich text content with WYSIWIG editor (Trix).
- URLs and views for storing attachments.
- Admin site integration.

Integrating Django Prose

To get started with Django Prose all you have to do is:

1. Install the `django-prose` package
2. Add `prose` in the `INSTALLED_APPS` setting
3. Use `Document` for partitioned and `RichTextField` for inline content

Integrating Django Prose with attachments

To get started with Django Prose all you have to do is:

1. Install the `django-prose` package
2. Add `prose` in the `INSTALLED_APPS` setting
3. Use `Document` for partitioned and `RichTextField` for inline content
4. Include `prose.urls` in your URLs

settings.py

```
# ...before
INSTALLED_APPS = [
    "django.contrib.admin",
    "django.contrib.auth",
    "django.contrib.contenttypes",
    "django.contrib.sessions",
    "django.contrib.messages",
    "django.contrib.staticfiles",
    "prose",
    "blog",
]
# ...after
```

models.py

```python
from django.conf import settings
from django.db import models

from prose.fields import RichTextField
from prose.models import AbstractDocument, Document


class Article(models.Model):
    title = models.CharField(max_length=255)
    author = models.ForeignKey(
        settings.AUTH_USER_MODEL,
        on_delete=models.CASCADE,
    )
    excerpt = RichTextField(blank=True, null=True)
    body = models.OneToOneField(Document, on_delete=models.CASCADE)
```

views.py

```
from django.shortcuts import render

from blog.models import Article

def blog_index(request):
    articles = Article.objects.all()
    context = {"articles": articles}
    return render(request, "blog/index.html", context)

def blog_article(request, pk):
    article = Article.objects.get(pk=pk)
    context = {"article": article}
    return render(request, "blog/article.html", context)
```

article.html

```
{% extends 'blog/base.html' %}

{% block title %}{{ article.title }} | Blog example | Django Prose{% endblock title %}
{% block description %}{{ article.excerpt }}{% endblock description %}

{% block content %}
<h1>{{ article.title }}</h1>
<div>{{ article.author.username }}</div>
<div>{{ article.body.content | safe }}</div>
{% endblock content %}
```

urls.py

```python
from django.conf import settings
from django.conf.urls.static import static
from django.contrib import admin
from django.urls import include, path

from blog import views


urlpatterns = [
    path("", views.blog_index, name="blog_index"),
    path("articles/<int:pk>/", views.blog_article, name="blog_article"),
    path("admin/", admin.site.urls),
    path("prose/", include("prose.urls")),
] + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

Django Prose vs `f"{you_name_it}"`

Django Prose vs `f"django-{wysiwig_editor}"`

It focuses on rich text workflows, instead of the editor component, unlike

- https://github.com/django-ckeditor/django-ckeditor
- https://github.com/islco/django-trix
- https://github.com/zakdoek/django-prosemirror

Django Prose vs content management libraries

It focuses on rich text integration in web apps, not full content management, unlike

- https://github.com/django-cms/django-cms
- https://github.com/wagtail/wagtail

The future

We have multiple ideas for the future of Django Prose

- Pluggable editor front-ends (e.g. Trix, ProseMirror, CKE etc.)
- Multiple output formats (e.g. HTML for e-mail, Markdown etc.)
- Attachments with pre-signed URLs
- Embedded content (e.g. Tweets, social media cards etc.)

Wrapping up

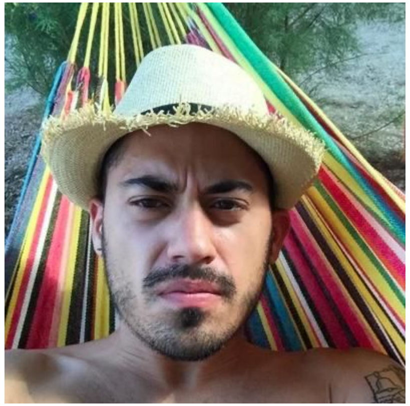
Django Prose

- GitHub: https://github.com/withlogicco/django-prose
- PyPI: https://pypi.org/project/django-prose/

whoami

- Paris Kasidiaris
- Co-founder at **LOGIC**

- https://twitter.com/pariskasid
- https://github.com/parisk
- https://www.linkedin.com/in/pkasid



LOGIC

- We provide professional services for web development with Django
- We publish Open Source at https://github.com/withlogicco
- Learn more at https://withlogic.co
- Get in touch at hey@withlogic.co

Thank you.

@pariskasid