django-staticinline Documentation

Release latest

Martin Mahner

Apr 29, 2023

CONTENTS

Full documentation: https://django-staticinline.readthedocs.io
Github Repository: https://github.com/bartTC/django-staticinline

CHAPTER

ONE

DJANGO-STATICINLINE

Works similar to Django's `static` templatetag, but this one includes the file directly in the template, rather than a link to it.

You can additionally post-process the file content using custom 'encoder'.

1.1 Compatibility Matrix:

Py/Dj	3.8	3.9	3.10	3.11
2.2 (LTS)	✓	✓	✓	✓
3.2 (LTS)	✓	✓	✓	✓
4.0	✓	✓	✓	✓
4.1	✓	✓	✓	✓
4.2 (LTS)	✓	✓	✓	✓

1.2 Quickstart

1. Put the StaticInlineAppConfig along your apps.

```
INSTALLED_APPS = [
    # ...
    'staticinline.apps.StaticInlineAppConfig',
]
```

2. Load the template tag and pass a filename as you'd do with a `static` template tag. You can also post-process the file content. In the example below we encode the content of the `mykey.pem` file with base64. Several encoders are already built-in, see the Encoder docs.

```
{% load staticinline %}

<style type="text/css">{% staticinline "myfile.css" %}</style>
My base64 encoded Key: {% staticinline "mykey.pem" encode="base64" cache=True %}
```

3. Enjoy the result:

```
<style type="text/css">body{ color: red; }</style>
My base64 encoded Key: LS0tIFN1cGVyIFByaXZhdGUgS2V5IC0tLQo=
```

1.3 Further reading:

1.3.1 Installation & Usage

Install the app with pip and add `staticinline.apps.StaticInlineAppConfig` to your installed apps in your settings.py:

```
pip install django-staticinline
```

```
INSTALLED_APPS = [
    # ...
    'staticinline.apps.StaticInlineAppConfig',
]
```

In a Django template load the `staticinline` templatetag and load a file using the same tag just as you'd load a file with the regular `static` templatetag.

```
{% load staticinline %}

<style type="text/css">{% staticinline "myfile.css" %}</style>
<script>{% staticinline "myfile.js" %}</script>
```

Note: If the file does not exist, and `DEBUG` is `False`, an empty string is returned and a error logfile is set. In case `DEBUG` is `True`, a `ValueError` is raised.

Caching

You can optioanlly cache the file content. This is particularly useful if you use expensive encoder processing.

Pass the `cache=True` argument. Additionally you can pass a timeout (in seconds) using the `cache_timeout` argument. If not set, the default timeout defined in the AppConfig is used.

```
{% load staticinline %}

{% staticinline "mykey.pem" encode="base64" cache=True %}
{% staticinline "mykey.pem" encode="base64" cache=True cache_timeout=3600 %}
```

1.3.2 Encoder: Transform files on the fly

You can automatically convert the file with the `encode` argument. django-staticinline ships with a couple of encoders:

List of build-in encoders:

base64

Transforms the file content to a base64 encoded string.

```
{% load staticinline %}
My key: {% staticinline "mykey.pem" encode="base64" %}
```

Becomes:

```
My key: LS0tIFN1cGVyIFByaXZhdGUgS2V5IC0tLQo=
```

---

data

Transforms the content into a data URI for use in CSS `url()` and HTML `src=""` attributes.

```
{% load staticinline %}
ul.checklist li.complete {
    background: url('{% staticinline "icons/check.png" encode="data" %}');
}
```

Becomes:

```
ul.checklist li.complete {
    background: url('\
goAAAANSUhEUgAAABAAAAAQAQMAAAAlPW0iAAAABlBMVEUAAAD//\
/+l2Z/dAAAAM0lEQVR4nGP4/5/h/1+G/58ZDrAz3D/McH8yw83ND\
DeNGe4Ug9C9zwz3gVLMDA/A6P9/AFGGFyjOXZtQAAAAAElFTkSuQ\
mCC');
}
```

---

sri

Generates a sha256 hash of the file content, suitable for Subresource Integrity verifications.

```
{% load staticfiles %}
{% load staticinline %}

<link rel="stylesheet"
      href="{% static "base.css" %}"
      integrity="{% staticinline "base.css" encode="sri" %}"
      crossorigin="anonymous"/>
```

Becomes:

```
<link rel="stylesheet"
      href="/static/base.css"
      integrity="sha256-aeB9jNF0zyjK656631roQQOsKgRocLazJdr6fmleg4I"
      crossorigin="anonymous"/>
```

Add a custom Encoder

You can add custom encoder by extending the AppConfig. See the Django docs for further information about AppConfig applications.

```
# myproject/apps.py
from staticinline.apps import StaticInlineAppConfig

class CustomStaticInlineAppConfig(StaticInlineAppConfig):

    # Add the custom 'upper' encoder to the list of build-in encoders.
    def get_encoder(self):
        encoder = super().get_encoder()
        encoder.update({
            'upper': self.encode_upper,
        })
        return encoder

    # Define the encoder itself. `data` contains the file content
    # and we transform all characters to uppercase here.
    def encode_upper(self, data, path):
        return data.upper()
```

In your `INSTALLED_APPS` setting you now point to your custom AppConfig:

```
# settings.py
INSTALLED_APPS = [
    # 'staticinline.apps.StaticInlineAppConfig',
    'myproject.apps.CustomStaticInlineAppConfig',
]
```

In a template you call it with the respective name:

```
{% load staticinline %}

{% staticinline "my-poem.txt" encode="upper" %}
```

1.3.3 Local Development & Testing

Local development is largely done with pipenv. Install the project once to get all necessary dependencies.

```
$ cd django-staticinline/

$ pip install pipenv  # If you don't have pipenv yet

$ pipenv install --dev
```

You can test the code against all currently support versions of Django and Python with tox by simply running `tox` in the project directorty:

```
$ pipenv run tox
```

A quicker way is to test the app against your current Django/Python version with pipenv directly:

```
$ pipenv run test
```

If you want to extend the documentation, you can compile it using pipenv as well:

```
$ pipenv run docs        # Compiles it once

$ pipenv run watch-docs  # Keeps a runserver-like process in the background,
                         # which compiles the docs on every file save.
```

1.3.4 Changelog

v1.4

- Django 3.2 to 4.2 compatibility and tests.
- Python 3.8 to 3.11 compatibility and tests.

v1.3 (2018-08-15)

- Added `cache` and `cache_timeout` templatetag arguments to store rendered values in cache.
- Added `data_response` AppConfig method to globally override the template tag response.

v1.2 (2018-08-14)

- Added support for Django 2.1 and Python 3.7.
- Added proper documentation.
- Added `sri` (Subresource Integrity) encoder to generate a sha256 for a given file.

v1.1 (2018-08-09)

- Added support for custom data encoders to modify file content on the fly.
- Added `data` and `base64` encoders, both convert data into base64.

v1.0 (2018-04-29)

- Initial release.