django-spa Documentation

Release 0.3.5

Dražen Lučanin

Dec 07, 2021

Contents

Contents:

CHAPTER 1

django-spa

Django package to serve a single-page app (SPA).

The following settings that make serving SPAs easier are handled in django-spa:

- *index.html* served when `/` requested
- all `/static/...` files served on `/...` as well
- Django's urls still work (Django admin, templates, Django REST framework APIs)
- everything else goes to `/` for frontend routing (e.g. react-router)

1.1 Usage

For an example of using django-spa to serve a create-react-app frontend that consumes a Django REST framework API, check out generator-django-rest.

As part of setting up django-spa, you also need to set up WhiteNoise, which we'll summarise here.

First, add `django-spa` to your *requirements.txt* and `pip install -r requirements.txt` (or `pipenv install django-spa`). Whitenoise is installed as a dependency, so no need to specify it extra.

Update *settings.py* with the Whitenoise & django-spa middleware:

```
MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'whitenoise.middleware.WhiteNoiseMiddleware',
    'spa.middleware.SPAMiddleware',
```

(continues on next page)

(continued from previous page)

```
    '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',
]
```

Disable runserver's static file serving by adding `runserver_nostatic` to the top of your INSTALLED_APPS list:

```
INSTALLED_APPS = [
    'whitenoise.runserver_nostatic',
    'django.contrib.staticfiles',
    # ...
]
```

Set the django-spa static file storage:

```
STATICFILES_STORAGE = 'spa.storage.SPAStaticFilesStorage'
```

You should be good to go!

1.2 Credits

Used some parts of the solution suggested in this WhiteNoise issue for serving index.html on `/`. This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

1.3 License

MIT

CHAPTER 2

Installation

2.1 Stable release

To install django-spa, run this command in your terminal:

```
$ pip install django-spa
```

This is the preferred method to install django-spa, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

2.2 From sources

The sources for django-spa can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/metakermit/django-spa
```

Or download the tarball:

```
$ curl  -OL https://github.com/metakermit/django-spa/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

CHAPTER 3

Usage

To use django-spa in a project:

```
import spa
```

CHAPTER 4

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

4.1 Types of Contributions

4.1.1 Report Bugs

Report bugs at https://github.com/metakermit/django-spa/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

4.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

4.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

4.1.4 Write Documentation

django-spa could always use more documentation, whether as part of the official django-spa docs, in docstrings, or even on the web in blog posts, articles, and such.

4.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/metakermit/django-spa/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

4.2 Get Started!

Ready to contribute? Here's how to set up *django-spa* for local development.

1. Fork the *django-spa* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/django-spa.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv django-spa
$ cd django-spa/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 spa tests
$ python setup.py test or py.test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

4.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2 and 3. Check https://travis-ci.org/metakermit/django-spa/pull_requests and make sure that the tests pass for all supported Python versions.

4.4 Tips

To run a subset of tests:

```
$ py.test tests.test_spa
```

4.5 Deploying

A reminder for the maintainers on how to deploy.

Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

```
$ bumpversion patch # possible: major / minor / patch
$ git push
$ git push --tags
```

Travis will then deploy to CheeseShop if tests pass.

CHAPTER 5

Indices and tables

- genindex
- modindex
- search