django-easywebpack Documentation

Release 0.2.1

Fndrz

Apr 19, 2018

Contents

Making Django and Webpack best friends.

CHAPTER 1

Contents:

1.1 Setup

1.1.1 Installation

At the command line:

```
$ pip install django-easywebpack
```

1.1.2 Configuration

Webpack

django-easywebpack requires the webpack-manifest-plugin.

To configure it, install webpack-manifest-plugin with yarn or npm:

```
$ yarn add webpack-manifest-plugin
```

Add webpack-manifest-plugin into `webpack.config.js`:

```
var ManifestPlugin = require('webpack-manifest-plugin');

const config = {

  ...

  plugins: [
    new ManifestPlugin()
  ]

  ...
```

```
}
```

Django

To use django-easywebpack, add it to your *INSTALLED_APPS*:

> Note: `'easywebpack'` must be placed before the staticfiles app for the management commands to work properly.

```
INSTALLED_APPS = (
    'easywebpack',
    ...
)
```

Then, configure it in your Django settings:

```
EASYWEBPACK = {
    'MANIFEST': 'path/to/manifest.json',
}
```

1.2 Usage

1.2.1 Template tags

The provided template tags must be loaded before they can be used:

```
{% load webpack_extras %}
```

webpack_include

`{% webpack_include filename %}`

Includes a file from the Webpack manifest. Only JS and CSS files are currently supported.

Example:

```
{% load webpack_extras app.js %}
```

1.2.2 Management commands

runserver

`$ django-admin runserver`

If `settings.DEBUG` is `True`, this runs webpack with `--env.development --mode=development`.

collectstatic

```
$ django-admin collectstatic
```

If `settings.DEBUG` is `True`, this runs webpack with `--env.development --mode=development`.

Otherwise, it runs webpack with `--env.production --mode=production`.

webpack

```
$ django-admin webpack
```

This command runs webpack. By default, it uses a `development` environment.

Options:

`--environment`

Selects a build environment. It can be set to `production` or `development`.

1.3 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:

1.3.1 Types of Contributions

Report Bugs

Report bugs at https://github.com/fndrz/django-easywebpack/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.

Fix Bugs

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

Implement Features

Look through the GitHub issues for features. Anything tagged with "feature" is open to whoever wants to implement it.

Write Documentation

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

Submit Feedback

The best way to send feedback is to file an issue at https://github.com/fndrz/django-easywebpack/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 :)

1.3.2 Get Started!

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

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

```
$ git clone git@github.com:<GitHub username>/django-easywebpack.git
```

3. Install requirements and create a development environment:

```
$ pip install pipenv
$ pipenv install --dev
$ pipenv shell
```

4. Create a branch for local development:

```
$ git checkout -b <bugfix/feature name>
```

   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:

```
$ pytest --flake8
$ pytest
$ tox
```

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

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

7. Submit a pull request through the GitHub website.

1.3.3 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 2.6, 2.7, and 3.3, and for PyPy. Check https://travis-ci.org/fndrz/django-easywebpack/pull_requests and make sure that the tests pass for all supported Python versions.

1.3.4 Tips

To run a subset of tests:

```
$ pytest <module>
```

1.3.5 Releasing

First, run punch to create a version update:

```
$ punch --part <major|minor|patch>
```

Once everything has been merged into master, locally publish the package:

```
$ python setup.py sdist bdist_wheel
$ twine upload dist/*
```

1.4 Credits

1.4.1 Development Lead

- Joshua Smith <thewanuki@gmail.com>
- Guy Jacks <guy.jacks@fndrz.com>

1.4.2 Contributors

None yet. Why not be the first?

1.5 History

1.5.1 0.2.1 (2018-04-19)

- Fix broken management commands.
- Add more information to setup.py.
- Integrate version updating with Punch.

1.5.2 0.2.0 (2018-04-18)

- First release on PyPI.