djangocms-redirect Documentation

Release 0.6.1.dev0

Nephila

Dec 05, 2020

Contents

CHAPTER 1

djangocms-redirect

A django CMS enabled application to handle redirects

This is heavily borrowed from `django.contrib.redirects` with three major changes:

- Selection of django CMS pages
- Selection of redirect status code
- Middleware can processed in the request or response phase

1.1 Why using process_request?

Doing database queries in the middleware `process_request` is heavily discouraged as it's a performance hit, especially when doing redirects which are just a tiny part of the processed requests. Except that sometimes it's just what you need (for example to "hide" content without deleting / unpublishing it) By caching both existing and non existing redirects for a given URL the performance hit is minimized for the use cases that requires `process_request`.

1.2 Documentation

The full documentation is at https://djangocms-redirect.readthedocs.io.

1.3 Installation

See https://djangocms-redirect.readthedocs.io/en/latest/installation.html

1.4 Features

- Set old and new path, by selection existing django CMS pages or writing down the complete address
- Select the redirect status code (301, 302)
- Support for status code 410

1.5 Credits

Tools used in rendering this package:

- Cookiecutter
- cookiecutter-djangopackage-helper

Contents:

1.5.1 Installation

- Install djangocms-redirect:

```
pip install djangocms-redirect
```

- Add to INSTALLED_APPS:

```
INSTALLED_APPS = [
    ...
    djangocms_redirect
    ...
]
```

- Add to MIDDLEWARE_CLASSES:

  If you intend to use `process_request` the add it near the top (after `django.middleware.cache.UpdateCacheMiddleware`); if using `process_response` add at the bottom (before `django.middleware.cache.FetchFromCacheMiddleware`):

```
MIDDLEWARE_CLASSES = [
    ...
    djangocms_redirect.middleware.RedirectMiddleware
    ...
]
```

- Choose if you want to process the redirect during the request (default) or response by setting:
  - `DJANGOCMS_REDIRECT_USE_REQUEST = True`: during request
  - `DJANGOCMS_REDIRECT_USE_REQUEST = False`: during response
- Migrate:

```
python manage.py migrate
```

The go to `http://mysite.com/admin/djangocms_redirect/` and create redirect instances.

Settings

- `DJANGOCMS_REDIRECT_USE_REQUEST`: If `True` the redirect check will be done in the request phase, to allow preempting any other logic. **Beware**: this will result in extra queries on **each** request because the redirects will be checked before the view logic triggers. If `False` the redirect will be checked in the response phase.
- `DJANGOCMS_REDIRECT_CACHE_TIMEOUT`: You can provide a custom cache timeout (Default: 3600 sec)
- `DJANGOCMS_REDIRECT_404_ONLY`: If `True` (the default) and `DJANGOCMS_REDIRECT_USE_REQUEST=False` the redirect will be checked only for responses that return 404 (the default `django.contrib.redirect` behavior). This is the lowest impact option in terms of performance and the advised configuration.

1.5.2 Usage

To manage redirect rules navigate to the `Redirect` section of the django admin (usually `http://www.yoursite.com/admin/djangocms_redirect/redirect`).

For each redirect you must provide:

- **Site**: The site for which you want to add a redirect.
- **Redirect from**: The path that need to be redirected: you can type any URL or select an existing django CMS page.
- **Redirect to**: The path to which the request will be redirected: you can type any URL or select an existing django CMS page.
- **Response code**: You can select 3 types of status_code header: 301 (permanent redirect), 302 (temporary redirect) or 410 (permanent unavailable resource).

Each **redirect from** URL must be unique and start with a slash. If you leave out the leading slash when creating a redirect, it is added automatically.

If the user requests a page without a trailing slash and there is no redirect for that URL but there is one for the URL with a trailing slash, that redirect is used. For backwards-compatibility, this is also true when `APPEND_SLASH=False`.

If `APPEND_SLASH=True` (the default), a trailing slash is added automatically when creating a redirect. That way, there is only ever a single relevant redirect, whether there is a trailing slash or not.

Subpath matching

Each redirect can match the exact incoming request path (the default behavior) or a subpath.

Subpath matching comes in two behaviour:

Plain subpath matching

The registered redirects will be checked against the incoming URL and the longest string matching the beginning of the request path will be selected.

The request will be then redirected by replacing the matching **Redirect from** with the **Redirect to** in the original URL.

Example

- Incoming request: `/en/some/path/`
- Redirect from: `/en/some`
- Redirect to: `/en/other`

- Resulting redirect: `/en/other/path/`

Catchall redirect

As in **plain subpath matching** the registered redirects will be checked against the incoming URL and the longest string matching the beginning of the request path will be selected.

The request will be then redirected to the **Redirect to** without further changes.

Example

- Incoming request: `/en/some/path/`
- Redirect from: `/en/some`
- Redirect to: `/en/other`
- Resulting redirect: `/en/other`

1.5.3 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:

Types of Contributions

Report Bugs

Report bugs at https://github.com/nephila/djangocms-redirect/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

djangocms-redirect could always use more documentation, whether as part of the official djangocms-redirect 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/nephila/djangocms-redirect/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 :)

Get Started!

Ready to contribute? Here's how to set up `djangocms-redirect` for local development.

1. Fork the `djangocms-redirect` repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/djangocms-redirect.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 djangocms-redirect
$ cd djangocms-redirect/
$ pip install -r requirements-test.txt
$ pip install -e .
```

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:

```
$ tox
```

To get tox, pip install it 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.

Development tips

This project allows you to use pre-commit to ensure an easy compliance to the project code styles.

If you want to use it, install it globally (for example with `pip3 install --user precommit`, but check installation instruction. When first cloning the project ensure you install the git hooks by running `pre-commit install`.

From now on every commit will be checked against our code style.

Check also the available tox environments with `tox -l`: the ones not marked with a python version number are tools to help you work on the project buy checking / formatting code style, running docs etc.

Testing tips

You can test your project using any specific combination of python, django and django cms.

For example `tox -py37-django30-cms37` runs the tests on python 3.7, Django 3.0 and django CMS 3.7.

Pull Request Guidelines

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

1. Pull request must be named with the following naming scheme:

   `<type>/(<optional-task-type>-)<number>-description`

   See below for available types.

2. The pull request should include tests.
3. If the pull request adds functionality, the docs should be updated. Documentation must be added in `docs` directory, and must include usage information for the end user. In case of public API method, add extended docstrings with full parameters description and usage example.
4. Add a changes file in `changes` directory describing the contribution in one line. It will be added automatically to the history file upon release. File must be named as `<issue-number>.<type>` with type being:

   - `.feature`: For new features.
   - `.bugfix`: For bug fixes.
   - `.doc`: For documentation improvement.
   - `.removal`: For deprecation or removal of public API.
   - `.misc`: For general issues.

   Check towncrier documentation for more details.

5. The pull request should work for all python / django / django CMS versions declared in tox.ini. Check the CI and make sure that the tests pass for all supported versions.

Release a version

1. Update authors file
2. Merge `develop` on `master` branch
3. Bump release via task: `inv tag-release (major|minor|patch)`
4. Update changelog via towncrier: `towncrier --yes`
5. Commit changelog with `git commit --amend` to merge with bumpversion commit
6. Create tag `git tag <version>`
7. Push tag to github
8. Publish the release from the tags page

9. If pipeline succeeds, push `master`
10. Merge `master` back on `develop`
11. Bump developement version via task: `inv tag-dev -l (major|minor|patch)`
12. Push `develop`

1.5.4 Credits

Development Lead

- Iacopo Spalletti <i.spalletti@nephila.it>

Previous maintainers

- Paolo Romolini <p.romolini@nephila.it>

Contributors

- Adam Chainz
- Gaetano D'Onghia <g.donghia@frankhood.it>
- Giovanni Bottalico <g.bottalico@frankhood.it>
- Nicola Ciccarone <n.ciccarone@frankhood.it>
- schroedingersZombie
- Vladimir Kuvandjiev
- Xiphoseer

1.5.5 History

0.6.0 (2020-11-15)

Features

- Drop Python 2, Django < 2.2 - Update toolchain (#39)
- Fix Handling of trailing slashes in redirects (#31)

Unreleased

- Nothing yet

0.5.0 (2019-12-27)

- Add compatibility with Django 2.2
- Drop compatibility with Django < 1.11
- Drop compatibility with django CMS < 3.6
- Move to django-app-helper
- Add support to match unquoted strings as redirect old path

0.4.0 (2019-08-22)

- Add subpath matching

0.3.1 (2019-07-13)

- Ignore querystring when matching redirect objects

0.3.0 (2019-03-11)

- Added compatibility to Django 2.0, 2.1

0.2.3 (unreleased)

- Add support to match unquoted strings as redirect old path

0.2.2 (2019-06-02)

- Ignore querystring when matching redirect objects

0.2.1 (2019-04-22)

- Fixed compatibility issue with Django 1.8

0.2.0 (2018-11-03)

- Updated for Django 1.11
- Added configurable cache timeout
- Added configuration option to check redirect on 404 only

0.1.1 (2017-11-19)

- Added missing migration.
- Fixed compatibility issue with Django 1.8

0.1.0 (2016-02-01)

- First release on PyPI.