Common helper libraries for Python Apps Documentation

Release 0.34.2

Scroll Tech

Sep 04, 2023

Contents:

Common helper libraries for Python Apps

Common helper libraries for Python Apps

- Free software: MIT license
- Documentation: https://apphelpers.readthedocs.io.

1.1 Features

- TODO

1.2 Credits

This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

Installation

2.1 Stable release

To install Common helper libraries for Python Apps, run this command in your terminal:

```
$ pip install apphelpers
```

This is the preferred method to install Common helper libraries for Python Apps, 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 Common helper libraries for Python Apps can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/scrolltech/apphelpers
```

Or download the tarball:

```
$ curl  -OL https://github.com/scrolltech/apphelpers/tarball/master
```

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

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

CHAPTER 3

Usage

To use Common helper libraries for Python Apps in a project:

```python
import apphelpers
```

3.1 Securing APIs

3.1.1 Example

```python
def foo_api():
    return "bar"

foo_api.login_required = True
```

3.1.2 Supported directives

login_required

Boolean: True / False

If this is set to True only authenticated users are allowed to access the API. By default APIs are public.

any_group_required

List/Tuple

User accessing the API must be member of any of the groups specified

all_groups_required

List/Tuple

User accessing the API must be member of all the groups specified

```python
def foo_api():
    return "bar"

foo_api.groups_required = ['admin', 'moderator']
```

groups_forbidden

List/Tuple

API Access is forbidden to the members of specified groups.

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/scrolltech/apphelpers/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

Common helper libraries for Python Apps could always use more documentation, whether as part of the official Common helper libraries for Python Apps 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/scrolltech/apphelpers/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 *apphelpers* for local development.

1. Fork the *apphelpers* repo on GitHub.

2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/apphelpers.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 apphelpers
$ cd apphelpers/
$ 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:

```
$ gunicorn tests.service:__hug_wsgi__
$ nosetests -sxv tests
```

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 3.6 and 3.7. Check https://travis-ci.org/scrolltech/apphelpers/pull_requests and make sure that the tests pass for all supported Python versions.

4.4 Tips

To run a subset of tests:

```
$ python -m unittest tests.test_apphelpers
```

4.5 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in CHANGELOG.rst). Then run:

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

Travis will then deploy to PyPI if tests pass.

Credits

5.1 Development Lead

- Scroll Tech <tech@scroll.in>

5.2 Contributors

None yet. Why not be the first?

History

6.1 0.34.1 (2023-09-01)

- Email INTERNAL_EMAIL_DOMAINS will also restrict bcc email recipients.

6.2 0.34.0 (2023-09-01)

- settings.INTERNAL_EMAIL_DOMAINS must me defined for email sending safety in non-prod env. e.g. IN-TERNAL_EMAIL_DOMAINS = ['example.com', 'example.org'] allows sending emails to only address ending with example.com or example.org

6.3 0.33.5 (2023-08-08)

- Reusable utility *format_msg* added in email module to format email message

6.4 0.33.4 (2023-08-08)

- Support for optionally addding headers added to email message

6.5 0.33.3 (2023-08-04)

- For site-bound sessions, restrict access if site_id is missing

6.6 0.33.2 (2023-07-20)

- session destroy fix for site-bound sessions

6.7 0.33.1 (2023-07-20)

- site_id check fix for site-bound sessions

6.8 0.33.0 (2023-05-23)

- Support for new directives: user_groups, user_site_groups, user_site_ctx
- Ambiguously named groups_required decorator is now replaced with any_group_required
- New decorator: all_groups_required

6.9 0.32.1 (2023-04-18)

- Support for resyncing & destroying context bound session

6.10 0.32.0 (2023-04-11)

- Support for context bound sessions

6.11 0.31.2 (2022-12-15)

- Implement ReadWriteCachedModel.decrement()

6.12 0.31.1 (2022-09-20)

- Extend sesion timeout fix for lookup key
- Implement sessions.sid2uid()

6.13 0.31.0 (2022-08-08)

- Rename Config directive MD_* to SMTP_*

6.14 0.21.1 (2022-06-09)

- Fix for SMTP+SSL connection

6.15 0.21.0 (2022-05-18)

- applogger: general purpose application logging

6.16 0.20.0 (2022-04-29)

- Support for custom authorizaion

6.17 0.19.1 (2021-10-07)

- Report function args in honeybadger context

6.18 0.9.2 (2019-05-20)

- New options *groups_forbidden* and *groups_required* to secure API access

6.19 0.1.0 (2019-03-24)

- First release on PyPI.

CHAPTER 7

Indices and tables

- genindex
- modindex
- search