django-purge Documentation

Release 1.0.0

Gregory N. Schmit

Nov 18, 2019

Contents:

Version: 1.0.0

CHAPTER 1

Models

class purge.models.**DatabasePurger**(**args*, ***kwargs*)

Represents a purging action on a set of database tables.

exception DoesNotExist

exception MultipleObjectsReturned

purge()

If this purger is enabled, purge the selected tables by age/quantity depending on the configuration.

classmethod purge_all()

Helper for running purge on all of the *DatabasePurger*'s.

selected_tables

Getter to display the selected tables in the admin UI.

class purge.models.**FilePurger**(*id*, *name*, *enabled*, *file_pattern*, *directory*, *recursive_search*, *delete_by_filename*, *filename_date_year_first*, *filename_date_day_first*, *delete_by_atime*, *delete_by_mtime*, *delete_by_ctime*, *age_in_days*)

exception DoesNotExist

exception MultipleObjectsReturned

filename_is_older_than(*filename*, *dt*)

Extract the datetime object from a filename and check it to see if it is too old.

purge(*directory=None*)

If this purger is enabled, evaluate the date and call the recursive purge method.

classmethod purge_all()

Helper for running purge on all of the *FilePurger*'s.

purge_recursive(*dt*, *directory=None*)

Purge files in this directory that match the criteria, and be recursive if that option is selected.

CHAPTER 2

Management Commands

class purge.management.commands.purge.**Command**(*stdout=None*, *stderr=None*, *no_color=False*, *force_color=False*)

handle(**args*, ***options*)
Run all database purgers

Documentation: https://django-purge.readthedocs.io

Source: https://github.com/gregschmit/django-purge

PyPI: https://pypi.org/project/django-purge/

Purge is a reusable Django app for regularly purging old database entries, like logs.

The Problem: Tables/models like sessions and logs can grow without limit.

The Solution: This app allows you to schedule database purging of old records. You can also make `FilePurgers` which can purge old files based on datestamps in the filename or timestamps in the meta-data (atime/mtime/ctime).

CHAPTER 3

How to Use

```
$ pip install django-purge
```

Include `purge` in your `INSTALLED_APPS`. Then, create your database purgers or file purgers in the admin interface.

Then, either periodically call the `purge` management command (e.g., via a system cronjob), or install and configure `django-cron` (add `purge.cron` to your `CRON_CLASSES` in your `settings.py`). The builtin `CronJob` class is set to run every 4 hours. You can change this by altering your `settings.py` and adding `PURGE_CRON_RUN_AT_TIMES` to an array of times you want to run the job at (e.g., `['1:00']` to run at 1am).

CHAPTER 4

Contributing

Create a pull request if you want to contribute. You must only contribute code that you have authored or otherwise hold the copyright to, and you must make any contributions to this project available under the MIT license.

To collaborators: don't push using the `--force` option.

CHAPTER 5

Dev Quickstart

Purge comes with a `settings.py` file, technically making it a Django project as well as a Django app. First clone, the repository into a location of your choosing:

```
$ git clone https://github.com/gregschmit/django-purge
```

Then you can go into the `django-purge` directory and do the initial migrations and run the server (you may need to type `python3` rather than `python`):

```
$ cd django-purge
$ python manage.py makemigrations
$ python manage.py migrate
$ python manage.py createsuperuser
...
$ python manage.py runserver
```

Then you can see the models at http://127.0.0.1:8000/admin.

CHAPTER 6

Indices and tables

- genindex
- modindex
- search

Python Module Index

p

Index