



django*celerymonitor* Documentation
Release 1.1.2

Jannis Leidel

Sep 27, 2017

Contents

1	About	3
2	History	5
3	Installation	7
4	Usage	9
5	Starting the monitoring process	11
6	Configuration	13
7	Contents	15
	Python Module Index	21

Version 1.1.2

Web <https://django-celery-monitor.readthedocs.io/>

Download https://pypi.python.org/pypi/django_celery_monitor

Source <https://github.com/jezdez/django-celery-monitor>

Keywords django, celery, events, monitoring

CHAPTER 1

About

This extension enables you to monitor Celery tasks and workers.

It defines two models (`django_celery_monitor.models.WorkerState` and `django_celery_monitor.models.TaskState`) used to store worker and task states and you can query this database table like any other Django model. It provides a Camera class (`django_celery_monitor.camera.Camera`) to be used with the Celery events command line tool to automatically populate the two models with the current state of the Celery workers and tasks.

CHAPTER 2

History

This package is a Celery 4 compatible port of the Django admin based monitoring feature that was included in the old [django-celery](#) package which is only compatible with Celery < 4.0. Other parts of [django-celery](#) were released as [django-celery-beat](#) (Database-backed Periodic Tasks) and [django-celery-results](#) (Celery result backends for Django).

CHAPTER 3

Installation

You can install `django_celery_monitor` either via the Python Package Index (PyPI) or from source.

To install using *pip*:

```
$ pip install -U django_celery_monitor
```


To use this with your project you need to follow these steps:

1. Install the `django_celery_monitor` library:

```
$ pip install django_celery_monitor
```

2. Add `django_celery_monitor` to `INSTALLED_APPS` in your Django project's `settings.py`:

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

Note that there is no dash in the module name, only underscores.

3. Create the Celery database tables by performing a database migrations:

```
$ python manage.py migrate django_celery_monitor
```

4. Go to the Django admin of your site and look for the “Celery Monitor” section.

Starting the monitoring process

To enable taking snapshots of the current state of tasks and workers you'll want to run the Celery events command with the appropriate camera class `django_celery_monitor.camera.Camera`:

```
$ celery -A proj events -l info --camera django_celery_monitor.camera.Camera --  
↪ frequency=2.0
```

For a complete listing of the command-line options available see:

```
$ celery events --help
```


CHAPTER 6

Configuration

There are a few settings that regulate how long the task monitor should keep state entries in the database. Either of the three should be a `datetime.timedelta` value or `None`.

- `monitor_task_success_expires` – Defaults to `timedelta(days=1)` (1 day)
The period of time to retain monitoring information about tasks with a `SUCCESS` result.
- `monitor_task_error_expires` – Defaults to `timedelta(days=3)` (3 days)
The period of time to retain monitoring information about tasks with an erroneous result (one of the following event states: `RETRY`, `FAILURE`, `REVOKED`).
- `monitor_task_pending_expires` – Defaults to `timedelta(days=5)` (5 days)
The period of time to retain monitoring information about tasks with a pending result (one of the following event states: `PENDING`, `RECEIVED`, `STARTED`, `REJECTED`, `RETRY`).

In your Celery configuration simply set them to override the defaults, e.g.:

```
from datetime import timedelta

monitor_task_success_expires = timedelta(days=7)
```


Copyright

django_celery_monitor User Manual

by Ask Solem and Jannis Leidel

Copyright © 2017, Jannis Leidel Copyright © 2016, Ask Solem

All rights reserved. This material may be copied or distributed only subject to the terms and conditions set forth in the [Creative Commons Attribution-ShareAlike 4.0 International](#) license.

You may share and adapt the material, even for commercial purposes, but you must give the original author credit. If you alter, transform, or build upon this work, you may distribute the resulting work only under the same license or a license compatible to this one.

Note: While the *django_celery_monitor documentation* is offered under the [Creative Commons Attribution-ShareAlike 4.0 International](#) license the *django_celery_monitor software* is offered under the [BSD License \(3 Clause\)](#)

API Reference

Release 1.1

Date Sep 27, 2017

`django_celery_monitor.camera`

The Celery events camera.

class `django_celery_monitor.camera.Camera(*args, **kwargs)`

The Celery events Polaroid snapshot camera.

TaskState

Return the data model to store task state in.

WorkerState

Return the data model to store worker state in.

expire_task_states

Return a twople of Celery task states and expiration timedeltas.

handle_task (*uuid_task*, *worker=None*)

Handle snapshotted event.

django_celery_monitor.humanize

Some helpers to humanize values.

`django_celery_monitor.humanize.naturaldate` (*date*, *include_seconds=False*)

Convert datetime into a human natural date string.

`django_celery_monitor.humanize.pluralize_day` (*n*)

Return a string with the number of days ago.

`django_celery_monitor.humanize.pluralize_month` (*n*)

Return a string with the number of months ago.

`django_celery_monitor.humanize.pluralize_week` (*n*)

Return a string with the number of weeks ago.

`django_celery_monitor.humanize.pluralize_year` (*n*)

Return a string with the number of years ago.

django_celery_monitor.managers

The model managers.

class `django_celery_monitor.managers.ExtendedQuerySet` (*model=None*, *query=None*, *using=None*, *hints=None*)

A custom model queryset that implements a few helpful methods.

select_for_update_or_create (*defaults=None*, ***kwargs*)

Extend `update_or_create` with `select_for_update`.

Look up an object with the given kwargs, updating one with defaults if it exists, otherwise create a new one. Return a tuple (object, created), where created is a boolean specifying whether an object was created.

This is a backport from Django 1.11 (<https://code.djangoproject.com/ticket/26804>) to support `select_for_update` when getting the object.

class `django_celery_monitor.managers.TaskStateQuerySet` (*model=None*, *query=None*, *using=None*, *hints=None*)

A custom model queryset for the TaskState model with some helpers.

active ()

Return all active task states.

expire_by_states (*states*, *expires*)

Expire task with one of the given states.

expired (*states*, *expires*)

Return all expired task states.

purge()

Purge all expired task states.

class django_celery_monitor.managers.**WorkerStateQuerySet** (*model=None, query=None, using=None, hints=None*)

A custom model queryset for the WorkerState model with some helpers.

django_celery_monitor.models

The data models for the task and worker states.

class django_celery_monitor.models.**TaskState** (**args, **kwargs*)

The data model to store the task state in.

args

The positional *task arguments*.

eta

An optional *datetime* describing the *ETA* for its processing.

expires

An optional *datetime* describing when the task *expires*.

hidden

Whether the task has been expired and will be purged by the event framework.

kwargs

The keyword *task arguments*.

name

The *task name*.

objects = <django.db.models.manager.ManagerFromTaskStateQuerySet object>

A TaskStateManager instance to query the *TaskState* model.

result

The result of the task.

retries

The number of retries.

runtime

The task runtime in seconds.

state

The *task state* as returned by Celery.

task_id

The task *UUID*.

traceback

The Python error traceback if raised.

tstamp

A *datetime* describing when the task was received.

worker

The worker responsible for the execution of the task.

class django_celery_monitor.models.**WorkerState** (**args, **kwargs*)

The data model to store the worker state in.

hostname

The hostname of the Celery worker.

is_alive()

Return whether the worker is currently alive or not.

last_heartbeat

A `datetime` describing when the worker was last seen.

objects = <django.db.models.manager.ManagerFromWorkerStateQuerySet object>

A `ExtendedManager` instance to query the `WorkerState` model.

django_celery_monitor.utils

Utilities.

`django_celery_monitor.utils.action(short_description, **kwargs)`

Set some admin action attributes.

`django_celery_monitor.utils.correct_awareness(value)`

Fix the given datetime timezone awareness.

`django_celery_monitor.utils.display_field(short_description, admin_order_field, allow_tags=True, **kwargs)`

Set some display_field attributes.

`django_celery_monitor.utils.fixedwidth(field, name=None, pt=6, width=16, maxlen=64, pretty=False)`

Render a field with a fixed width.

`django_celery_monitor.utils.fromtimestamp(value)`

Return an aware or naive datetime from the given timestamp.

`django_celery_monitor.utils.make_aware(value)`

Make the given datetime aware of a timezone.

Change history

release-date 2017-05-18 11:30 a.m. UTC+2

release-by Jannis Leidel

- More packaging fixes. Sigh.

release-date 2017-05-18 10:30 a.m. UTC+2

release-by Jannis Leidel

- Fix the folder that the extra stylesheet file was stored in.

release-date 2017-05-11 10:25 p.m. UTC+2

release-by Jannis Leidel

- Use `SELECT FOR UPDATE` SQL statements for updating the task and worker states to improve resiliance against race conditions by multiple simultaneously running cameras.
- Move worker state cache from in-process dictionary into database side timestamp to decide whether to do another worker update or not.
- Improved code structure by moving all utilities into same module.

release-date 2017-05-08 16:05 a.m. UTC+2

release-by Jannis Leidel

- Import Django models inline to prevent import time side effect.
- Run `django.setup()` when installing the Camera.

release-date 2017-05-03 10:17 a.m. UTC+2

release-by Jannis Leidel

- Fix the Python package manifest.
- Fix README rendering.

1.0.0

release-date 2017-05-03 08:35 a.m. UTC+2

release-by Jannis Leidel

- Initial release by extracting the monitor code from the old `django-celery` app.
- Add ability to override the expiry timedelta for the task monitor via the Celery configuration.
- Add Python 3.6 and Django 1.11 to test matrix. Supported versions of Django 1.8 LTS, 1.9, 1.10 and 1.11 LTS. Supported versions of Python are 2.7, 3.4, 3.5 and 3.6 (for Django 1.11).

d

`django_celery_monitor.camera`, [15](#)
`django_celery_monitor.humanize`, [16](#)
`django_celery_monitor.managers`, [16](#)
`django_celery_monitor.models`, [17](#)
`django_celery_monitor.utils`, [18](#)

A

action() (in module django_celery_monitor.utils), 18
 active() (django_celery_monitor.managers.TaskStateQuerySet
 method), 16
 args (django_celery_monitor.models.TaskState attribute),
 17

C

Camera (class in django_celery_monitor.camera), 15
 correct_awareness() (in module
 django_celery_monitor.utils), 18

D

display_field() (in module django_celery_monitor.utils),
 18
 django_celery_monitor.camera (module), 15
 django_celery_monitor.humanize (module), 16
 django_celery_monitor.managers (module), 16
 django_celery_monitor.models (module), 17
 django_celery_monitor.utils (module), 18

E

eta (django_celery_monitor.models.TaskState attribute),
 17
 expire_by_states() (django_celery_monitor.managers.TaskStateQuerySet
 method), 16
 expire_task_states (django_celery_monitor.camera.Camera
 attribute), 16
 expired() (django_celery_monitor.managers.TaskStateQuerySet
 method), 16
 expires (django_celery_monitor.models.TaskState at-
 tribute), 17
 ExtendedQuerySet (class in
 django_celery_monitor.managers), 16

F

fixedwidth() (in module django_celery_monitor.utils), 18
 fromtimestamp() (in module
 django_celery_monitor.utils), 18

H

handle_task() (django_celery_monitor.camera.Camera
 method), 16
 hidden (django_celery_monitor.models.TaskState at-
 tribute), 17
 hostname (django_celery_monitor.models.WorkerState
 attribute), 17

I

is_alive() (django_celery_monitor.models.WorkerState
 method), 18

K

kwargs (django_celery_monitor.models.TaskState at-
 tribute), 17

L

last_heartbeat (django_celery_monitor.models.WorkerState
 attribute), 18

M

make_aware() (in module django_celery_monitor.utils),
 18

N

name (django_celery_monitor.models.TaskState at-
 tribute), 17
 naturaldate() (in module
 django_celery_monitor.humanize), 16

O

objects (django_celery_monitor.models.TaskState at-
 tribute), 17
 objects (django_celery_monitor.models.WorkerState at-
 tribute), 18

P

pluralize_day() (in module
 django_celery_monitor.humanize), 16

pluralize_month() (in module
django_celery_monitor.humanize), 16
pluralize_week() (in module
django_celery_monitor.humanize), 16
pluralize_year() (in module
django_celery_monitor.humanize), 16
purge() (django_celery_monitor.managers.TaskStateQuerySet
method), 16

R

result (django_celery_monitor.models.TaskState at-
tribute), 17
retries (django_celery_monitor.models.TaskState at-
tribute), 17
runtime (django_celery_monitor.models.TaskState
attribute), 17

S

select_for_update_or_create()
(django_celery_monitor.managers.ExtendedQuerySet
method), 16
state (django_celery_monitor.models.TaskState attribute),
17

T

task_id (django_celery_monitor.models.TaskState at-
tribute), 17
TaskState (class in django_celery_monitor.models), 17
TaskState (django_celery_monitor.camera.Camera
attribute), 15
TaskStateQuerySet (class in
django_celery_monitor.managers), 16
traceback (django_celery_monitor.models.TaskState at-
tribute), 17
tstamp (django_celery_monitor.models.TaskState at-
tribute), 17

W

worker (django_celery_monitor.models.TaskState at-
tribute), 17
WorkerState (class in django_celery_monitor.models), 17
WorkerState (django_celery_monitor.camera.Camera at-
tribute), 16
WorkerStateQuerySet (class in
django_celery_monitor.managers), 17