django-celery Documentation

Release 2.5.3

Ask Solem

Nov 16, 2017

Contents

Contents:

CHAPTER 1

django-celery - Celery Integration for Django

Version 2.5.3

Web http://celeryproject.org/

Download http://pypi.python.org/pypi/django-celery/

Source http://github.com/ask/django-celery/

Keywords celery, task queue, job queue, asynchronous, rabbitmq, amqp, redis, python, django, webhooks, queue, distributed

–

django-celery provides Celery integration for Django; Using the Django ORM and cache backend for storing results, autodiscovery of task modules for applications listed in `INSTALLED_APPS`, and more.

1.1 Using django-celery

To enable `django-celery` for your project you need to add `djcelery` to `INSTALLED_APPS`:

```
INSTALLED_APPS += ("djcelery", )
```

then add the following lines to your `settings.py`:

```
import djcelery
djcelery.setup_loader()
```

Everything works the same as described in the Celery User Manual, except you need to invoke the programs through `manage.py`:

Program	Replace with
`celeryd`	`python manage.py celeryd`
`celeryctl`	`python manage.py celeryctl`
`celerybeat`	`python manage.py celerybeat`
`camqadm`	`python manage.py camqadm`
`celeryev`	`python manage.py celeryev`
`celeryd-multi`	`python manage.py celeryd_multi`

The other main difference is that configuration values are stored in your Django projects' `settings.py` module rather than in `celeryconfig.py`.

If you're trying celery for the first time you should start by reading Getting started with django-celery

1.1.1 Special note for mod_wsgi users

If you're using `mod_wsgi` to deploy your Django application you need to include the following in your `.wsgi` module:

```
import djcelery
djcelery.setup_loader()
```

1.2 Documentation

The Celery User Manual contains user guides, tutorials and an API reference. Also the django-celery documentation, contains information about the Django integration.

1.3 Installation

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

To install using `pip`,:

```
$ pip install django-celery
```

To install using `easy_install`,:

```
$ easy_install django-celery
```

You will then want to create the necessary tables. If you are using south for schema migrations, you'll want to:

```
$ python manage.py migrate djcelery
```

For those who are not using south, a normal **`syncdb`** will work:

```
$ python manage.py syncdb
```

1.3.1 Downloading and installing from source

Download the latest version of `django-celery` from http://pypi.python.org/pypi/django-celery/

You can install it by doing the following,:

```
$ tar xvfz django-celery-0.0.0.tar.gz
$ cd django-celery-0.0.0
# python setup.py install # as root
```

1.3.2 Using the development version

You can clone the git repository by doing the following:

```
$ git clone git://github.com/ask/django-celery.git
```

1.4 Getting Help

1.4.1 Mailing list

For discussions about the usage, development, and future of celery, please join the celery-users mailing list.

1.4.2 IRC

Come chat with us on IRC. The #celery channel is located at the Freenode network.

1.5 Bug tracker

If you have any suggestions, bug reports or annoyances please report them to our issue tracker at http://github.com/ask/django-celery/issues/

1.6 Wiki

http://wiki.github.com/ask/celery/

1.7 Contributing

Development of `django-celery` happens at Github: http://github.com/ask/django-celery

You are highly encouraged to participate in the development. If you don't like Github (for some reason) you're welcome to send regular patches.

1.8 License

This software is licensed under the `New BSD License`. See the `LICENSE` file in the top distribution directory for the full license text.

CHAPTER 2

Getting Started

2.1 First steps with Django

This document has been moved into the main Celery documentation, you can find it at:;

http://ask.github.com/celery/django/first-steps-with-django.html

CHAPTER 3

Frequently Asked Questions

3.1 Generating a template in a task doesn't seem to respect my i18n settings?

Answer: To enable the Django translation machinery you need to activate it with a language. **Note**: Be sure to reset to the previous language when done.

```
>>> from django.utils import translation
```

```
>>> prev_language = translation.get_language()
>>> translation.activate(language)
>>> try:
...     render_template()
... finally:
        translation.activate(prev_language)
```

The common pattern here would be for the task to take a `language` argument:

```
from celery.decorators import task

from django.utils import translation
from django.template.loader import render_to_string

@task()
def generate_report(template="report.html", language=None):
    prev_language = translation.get_language()
    language and translation.activate(language)
    try:
        report = render_to_string(template)
    finally:
        translation.activate(prev_language)
    save_report_somewhere(report)
```

3.2 The celery test-suite is failing

Answer: If you're running tests from your Django project, and the celery test suite is failing in that context, then follow the steps below. If the celery tests are failing in another context, please report an issue to our issue tracker at GitHub:

http://github.com/ask/celery/issues/

That Django is running tests for all applications in `INSTALLED_APPS` by default is a pet peeve for many. You should use a test runner that either

1. Explicitly lists the apps you want to run tests for, or
2. Make a test runner that skips tests for apps you don't want to run.

For example the test runner that celery is using:

http://github.com/ask/celery/blob/f90491fe0194aa472b5aecdefe5cc83289e65e69/celery/tests/runners.py

To use this test runner, add the following to your `settings.py`:

```
TEST_RUNNER = "djcelery.tests.runners.CeleryTestSuiteRunner",
TEST_APPS = (
    "app1",
    "app2",
    "app3",
    "app4",
)
```

Or, if you just want to skip the celery tests:

```
INSTALLED_APPS = (.....)
TEST_RUNNER = "djcelery.tests.runners.CeleryTestSuiteRunner",
TEST_APPS = filter(lambda k: k != "celery", INSTALLED_APPS)
```

CHAPTER 4

Cookbook

4.1 Unit Testing

4.1.1 Testing with Django

The first problem you'll run in to when trying to write a test that runs a task is that Django's test runner doesn't use the same database as your celery daemon is using. If you're using the database backend, this means that your tombstones won't show up in your test database and you won't be able to get the return value or check the status of your tasks.

There are two ways to get around this. You can either take advantage of `CELERY_ALWAYS_EAGER = True` to skip the daemon, or you can avoid testing anything that needs to check the status or result of a task.

4.1.2 Using a custom test runner to test with celery

If you're going the `CELERY_ALWAYS_EAGER` route, which is probably better than just never testing some parts of your app, a custom Django test runner does the trick. Celery provides a simple test runner, but it's easy enough to roll your own if you have other things that need to be done. http://docs.djangoproject.com/en/dev/topics/testing/#defining-a-test-runner

For this example, we'll use the `djcelery.contrib.test_runner` to test the `add` task from the User Guide: Tasks examples in the Celery documentation.

To enable the test runner, set the following settings:

```
TEST_RUNNER = 'djcelery.contrib.test_runner.CeleryTestSuiteRunner'
```

Then we can put the tests in a `tests.py` somewhere:

```
from django.test import TestCase
from myapp.tasks import add

class AddTestCase(TestCase):

```

```
    def testNoError(self):
        """Test that the ``add`` task runs with no errors,
        and returns the correct result."""
        result = add.delay(8, 8)

        self.assertEquals(result.get(), 16)
        self.assertTrue(result.successful())
```

This test assumes that you put your example `add` task in `maypp.tasks` so adjust the import for wherever you put the class.

This page contains common recipes and techniques.

CHAPTER 5

API Reference

Release 2.5

Date Nov 16, 2017

5.1 App - djcelery.app

djcelery.app.**app = None**
The Django-Celery app instance.

5.2 Views - djcelery.views

5.3 URLs - djcelery.urls

5.4 Django Models - celery.models

TASK_STATUS_PENDING
The string status of a pending task.

TASK_STATUS_RETRY
The string status of a task which is to be retried.

TASK_STATUS_FAILURE
The string status of a failed task.

TASK_STATUS_DONE
The string status of a task that was successfully executed.

TASK_STATUSES
List of possible task statuses.

TASK_STATUSES_CHOICES
Django tuple of possible values for the task statuses, for usage in model/form fields `choices` argument.

class TaskMeta
Model for storing the result and status of a task.

Note Only used if you're running the `database` backend.

task_id
The unique task id.

status
The current status for this task.

result
The result after successful/failed execution. If the task failed, this contains the execption it raised.

date_done
The date this task changed status.

class PeriodicTaskMeta
Metadata model for periodic tasks.

name
The name of this task, as registered in the task registry.

last_run_at
The date this periodic task was last run. Used to find out when it should be run next.

total_run_count
The number of times this periodic task has been run.

task
The class/function for this task.

delay()
Delay the execution of a periodic task, and increment its total run count.

5.5 Managers - djcelery.managers

5.6 Celery Loaders - djcelery.loaders

5.7 Periodic Task Schedulers - djcelery.schedulers

5.8 Event Snapshots - djcelery.snapshot

5.9 Database Backend - djcelery.backends.database

5.10 Cache Backend - djcelery.backends.cache

5.11 Contrib: Test Runner - djcelery.contrib.test_runner

5.12 Humanize utils - djcelery.humanize

5.13 Utilities - djcelery.utils

djcelery.utils.**make_aware**(*value*)

djcelery.utils.**make_naive**(*value*)

djcelery.utils.**now**()

CHAPTER 6

Change history

6.1 2.5.3

release-date 2012-04-13 06:16 P.M GMT

by Ask Solem

- 2.5.2 release broke installation because of an import in the package.

  Fixed by not having setup.py import the djcelery module anymore, but rather parsing the package file for metadata.

6.2 2.5.2

release-date 2012-04-13 05:00 P.M GMT

by Ask Solem

- PeriodicTask admin now lists the enabled field in the list view

  Contributed by Gabe Jackson.

- Fixed a compatibility issue with Django < 1.3

  Fix contributed by Roman Barczyski

- Admin monitor now properly escapes args and kwargs.

  Fix contributed by Serj Zavadsky

- PeriodicTask admin now gives error if no schedule set (or both set) (Issue #126).
- examples/demoproject has been updated to use the Django 1.4 template.
- Database connection is no longer closed for eager tasks (Issue #116).

  Fix contributed by Mark Lavin.

- The first-steps document for django-celery has been moved to the main Celery documentation.
- djcelerymon command no longer worked properly, this has now been fixed (Issue #123).

6.3 2.5.1

release-date 2012-03-01 01:00 P.M GMT

by Ask Solem

6.3.1 Fixes

- Now depends on Celery 2.5.1
- Fixed problem with recursive imports when USE_I18N was enabled (Issue #109).
- The `CELERY_DB_REUSE_MAX` setting was not honored.
- The djcelerymon command no longer runs with DEBUG.

  To enable debug you can set the `DJCELERYMON_DEBUG` environment variable.
- Fixed eventlet/gevent compatability with Django 1.4's new thread sharing detection.
- Now depends on django-picklefield 0.2.0 or greater.

  Previous versions would not work correctly with Django 1.4.

6.4 2.5.0

release-date 2012-02-24 02:00 P.M GMT

by Ask Solem

6.4.1 Important Notes

- Now depends on Celery 2.5.
- Database schema has been updated.

  After upgrading you need migrate using South, or migrate manually as described below.

  These changes means that expiring results will be faster and take less memory than before.

  In addition a description field to the PeriodicTask model has been added so that the purpose of a periodic task in the database can be documented via the Admin interface.

  South Migration

  To migrate using South execute the following command:

  ```
  $ python manage.py migrate djcelery
  ```

  If this is a new project that is also using South then you need to fake the migration:

  $ python manage.y migrate djcelery –fake

  Manual Migration

  To manually add the new fields,

  using PostgreSQL:

  using MySQL:

```
ALTER TABLE celery_taskmeta
    ADD hidden TINYINT NOT NULL DEFAULT 0;

ALTER TABLE celery_tasksetmeta
    ADD hidden TINYINT NOT NULL DEFAULT 0;

ALTER TABLE djcelery_periodictask
    ADD description TEXT NOT NULL DEFAULT "";
```

using SQLite:

```
ALTER TABLE celery_taskmeta
    ADD hidden BOOL NOT NULL DEFAULT FALSE;
ALTER TABLE celery_tasksetmeta
    ADD hidden BOOL NOT NULL DEFAULT FALSE;
ALTER TABLE djcelery_periodictask
    ADD description VARCHAR(200) NOT NULL DEFAULT "";
```

6.4.2 News

- Auto-discovered task modules now works with the new auto-reloader functionality.
- The database periodic task scheduler now tried to recover from operational database errors.
- The periodic task schedule entry now accepts both int and timedelta (Issue #100).
- 'Connection already closed' errors occurring while closing the database connection are now ignored (Issue #93).
- The `djcelerymon` command used to start a Django admin monitor instance outside of Django projects now starts without a celery config module.
- Should now work with Django 1.4's new timezone support.

  Contributed by Jannis Leidel and Donald Stufft.
- South migrations did not work properly.

  Fix contributed by Christopher Grebs.
- celeryd-multi now preserves django-related arguments, like `--settings` (Issue #94).
- Migrations now work with Django < 1.3 (Issue #92).

  Fix contributed by Jude Nagurney.
- The expiry of the database result backend can now be an int (Issue #84).

6.5 2.4.2

release-date 2011-11-14 12:00 P.M GMT

- Fixed syntax error in South migrations code (Issue #88).

  Fix contributed by Olivier Tabone.

6.6 2.4.1

release-date 2011-11-07 06:00 P.M GMT

by Ask Solem

- Management commands was missing command line arguments because of recent changes to Celery.
- Management commands now supports the `--broker|-b` option.
- South migrations now ignores errors when tables already exist.

6.7 2.4.0

release-date 2011-11-04 04:00 P.M GMT

by Ask Solem

6.7.1 Important Notes

This release adds South migrations, which well assist users in automatically updating their database schemas with each django-celery release.

6.7.2 News

- Now depends on Celery 2.4.0 or higher.
- South migrations have been added.

  Migration 0001 is a snapshot from the previous stable release (2.3.3). For those who do not use South, no action is required. South users will want to read the *Upgrading for south users* section below.

  Contributed by Greg Taylor.
- Test runner now compatible with Django 1.4.

  Test runners are now classes instead of functions, so you have to change the `TEST_RUNNER` setting to read:

  ```
  TEST_RUNNER = "djcelery.contrib.test_runner.CeleryTestSuiteRunner"
  ```

  Contributed by Jonas Haag.

6.7.3 Upgrading for south users

For those that are already using django-celery 2.3.x, you'll need to fake the newly added migration 0001, since your database already has the current `djcelery_*` and `celery_*` tables:

```
$ python manage.py migrate djcelery 0001 --fake
```

If you're upgrading from the 2.2.x series, you'll want to drop/reset your `celery_*` and `djcelery_*` tables and run the migration:

```
$ python manage.py migrate djcelery
```

6.8 2.3.3

release-date 2011-08-22 12:00 AM BST

- Precedence issue caused database backend tables to not be created (Issue #62).

6.9 2.3.2

release-date 2011-08-20 12:00 AM BST

- Fixes circular import of DatabaseBackend.

6.10 2.3.1

release-date 2011-08-11 12:00 PM BST

- Django database result backend tables were not created.

  If you are having troubles because of this, be sure you do a `syncdb` after upgrading, that should resolve the issue.

6.11 2.3.0

release-date 2011-08-05 12:00 PM BST

- Now depends on Celery 2.3.0

  Please read the Celery 2.3.0 changelog!

6.12 2.2.4

- celerybeat: DatabaseScheduler would not react to changes when using MySQL and the default transaction isolation level `REPEATABLE-READ` (Issue #41).

  It is still recommended that you use isolation level `READ-COMMITTED` (see the Celery FAQ).

6.13 2.2.3

release-date 2011-02-12 16:00 PM CET

- celerybeat: DatabaseScheduler did not respect the disabled setting after restart.
- celeryevcam: Expiring objects now works on PostgreSQL.
- Now requires Celery 2.2.3

6.14 2.2.2

release-date 2011-02-03 16:00 PM CET

- Now requires Celery 2.2.2
- Periodic Task Admin broke if the CELERYBEAT_SCHEDULE setting was not set.
- DatabaseScheduler No longer creates duplicate interval models.
- The djcelery admin templates were not included in the distribution.

6.15 2.2.1

release-date 2011-02-02 16:00 PM CET

- Should now work with Django versions previous to 1.2.

6.16 2.2.0

release-date 2011-02-01 10:00 AM CET

- Now depends on Celery v2.2.0
- djceleryadm: Adds task actions Kill and Terminate task
- celerycam: Django's queryset.delete() fetches everything in memory THEN deletes, so we need to use raw SQL to expire objects.
- djcelerymon: Added Command.stdout + Command.stderr (Issue #23).
- Need to close any open database connection after any embedded celerybeat process forks.
- Added contrib/requirements/py25.txt
- Demoproject now does `djcelery.setup_loader` in settings.py.

6.17 2.1.1

release-date 2010-10-14 02:00 PM CEST

- Now depends on Celery v2.1.1.
- Snapshots: Fixed bug with losing events.
- Snapshots: Limited the number of worker timestamp updates to once every second.
- Snapshot: Handle transaction manually and commit every 100 task updates.
- snapshots: Can now configure when to expire task events.

  New settings:

  - `CELERYCAM_EXPIRE_SUCCESS` (default 1 day),
  - `CELERYCAM_EXPIRE_ERROR` (default 3 days), and
  - `CELERYCAM_EXPIRE_PENDING` (default 5 days).

- Snapshots: `TaskState.args` and `TaskState.kwargs` are now represented as `TextField` instead of `CharField`.

  If you need to represent arguments larger than 200 chars you have to migrate the table.

- `transaction.commit_manually` doesn't accept arguments on older Django version.

  Should now work with Django versions previous to v1.2.

- The tests doesn't need `unittest2` anymore if running on Python 2.7.

6.18 2.1.0

release-date 2010-10-08 12:00 PM CEST

6.18.1 Important Notes

This release depends on Celery version 2.1.0. Be sure to read the Celery changelog before you upgrade: http://ask.github.com/celery/changelog.html#version-2-1-0

6.18.2 News

- The periodic task schedule can now be stored in the database and edited via the Django Admin interface.

  To use the new database schedule you need to start `celerybeat` with the following argument:

```
$ python manage.py celerybeat -S djcelery.schedulers.DatabaseScheduler
```

  Note that you need to add your old periodic tasks to the database manually (using the Django admin interface for example).

- New Celery monitor for the Django Admin interface.

  To start monitoring your workers you have to start your workers in event mode:

```
$ python manage.py celeryd -E
```

  (you can do this without restarting the server too:

```
>>> from celery.task.control import broadcast
>>> broadcast("enable_events")
```

  You need to do a syncdb to create the new tables:

      python manage.py syncdb

  Then you need to start the snapshot camera:

```
$ python manage.py celerycam -f 2.0
```

  This will take a snapshot of the events every 2 seconds and store it in the database.

6.18.3 Fixes

- database backend: Now shows warning if polling results with transaction isolation level repeatable-read on MySQL.

  See http://github.com/ask/django-celery/issues/issue/6

- database backend: get result does no longer store the default result to database.

  See http://github.com/ask/django-celery/issues/issue/6

6.19 2.0.2

6.19.1 Important notes

- Due to some applications loading the Django models lazily, it is recommended that you add the following lines to your `settings.py`:

```
    import djcelery
    djcelery.setup_loader()

This will ensure the Django celery loader is set even though the
model modules haven't been imported yet.
```

6.19.2 News

- `djcelery.views.registered_tasks`: Added a view to list currently known tasks.

6.20 2.0.0

release-date 2010-07-02 02:30 P.M CEST

- Initial release

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

Python Module Index

d

Index