invenio-celery Documentation

Release 1.0.1

CERN

Dec 06, 2018

Contents

Celery distributed task queue module for Invenio.

Invenio-Celery is a small discovery layer that takes care of discovering and loading tasks from other Invenio modules, as well as providing configuration defaults for Celery usage in Invenio. Invenio-Celery relies on Flask-CeleryExt for integrating Flask and Celery with application factories.

Further documentation is available on https://invenio-celery.readthedocs.io/

CHAPTER 1

User's Guide

This part of the documentation will show you how to get started in using Invenio-Celery.

1.1 Installation

Invenio-Celery is on PyPI so all you need is:

```
$ pip install invenio-celery
```

1.2 Configuration

Default configuration values for Celery integration.

For further Celery configuration variables see Celery documentation.

`invenio_celery.config.`**`CELERY_ACCEPT_CONTENT = ['json', 'msgpack', 'yaml']`**
A whitelist of content-types/serializers.

`invenio_celery.config.`**`CELERY_BROKER_URL = 'redis://localhost:6379/0'`**
Broker settings.

`invenio_celery.config.`**`CELERY_RESULT_BACKEND = 'redis://localhost:6379/1'`**
The backend used to store task results.

`invenio_celery.config.`**`CELERY_RESULT_SERIALIZER = 'msgpack'`**
Result serialization format. Default is `msgpack`.

`invenio_celery.config.`**`CELERY_TASK_SERIALIZER = 'msgpack'`**
The default serialization method to use. Default is `msgpack`.

1.3 Usage

Celery distributed task queue module for Invenio.

Invenio-Celery is a small discovery layer that takes care of discovering and loading tasks from other Invenio modules, as well as providing configuration defaults for Celery usage in Invenio. Invenio-Celery relies on Flask-CeleryExt for integrating Flask and Celery with application factories.

1.3.1 Defining tasks

Invenio modules that wish to define Celery tasks should use the `@shared_task` decorator (usually in `tasks.py`):

```
# mymodule/tasks.py
from celery import shared_task

@shared_task
def sum(x, y):
     return x + y
```

Additionally the Invenio module should add the task module into the `invenio_celery.tasks` entry point:

```
# setup.py
setup(
    # ...
    entry_points=[
        'invenio_celery.tasks' : [
            'mymodule = mymodule.tasks'
        ]
    ]
)
```

1.3.2 Using tasks

Invenio modules that need to call tasks do not need to do anything special as long as the Invenio-Celery extension has been initialized. Hence calling tasks is as simple as:

```
from mymoudle.tasks import sum
result = sum.delay(2, 2)
```

1.3.3 Periodic tasks

Periodic tasks can be configured via `CELERYBEAT_SCHEDULE` configuration variable:

```
# config.py
CELERYBEAT_SCHEDULE = {
    'indexer': {
        'task': 'invenio_indexer.tasks.process_bulk_queue',
        'schedule': timedelta(minutes=5),
    },
}
```

For further information about see Periodic Tasks chapter of the Celery documentation.

1.3.4 Celery workers

Invenio-Celery hooks into the Celery application loading process so that when a worker starts, all the tasks modules defined in `invenio_celery.tasks` will be imported and cause the tasks to be registered in the worker. Note that this only happens on the Celery worker side which needs to know upfront all the possible tasks.

For further details on how to setup Celery and define an Celery application factory please see Flask-CeleryExt

CHAPTER 2

API Reference

If you are looking for information on a specific function, class or method, this part of the documentation is for you.

2.1 API Docs

Celery application for Invenio.

class `invenio_celery.ext.`**`InvenioCelery`**(*app=None*, ***kwargs*)

Invenio celery extension.

Extension initialization.

`disable_queue`(*name*)
: Disable given Celery queue.

`enable_queue`(*name*)
: Enable given Celery queue.

`get_active_tasks`()
: Return a list of UUIDs of active tasks.

`get_queues`()
: Return a list of current active Celery queues.

`init_app`(*app*, *entry_point_group='invenio_celery.tasks'*, ***kwargs*)
: Initialize application object.

`init_config`(*app*)
: Initialize configuration.

`load_entry_points`()
: Load tasks from entry points.

`suspend_queues`(*active_queues*, *sleep_time=10.0*)
: Suspend Celery queues and wait for running tasks to complete.

`invenio_celery.ext.`**`celery_module_imports`**(*sender*, *signal=None*, ***kwargs*)

Load shared celery tasks.

CHAPTER 3

Additional Notes

Notes on how to contribute, legal information and changes are here for the interested.

3.1 Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

3.1.1 Types of Contributions

Report Bugs

Report bugs at https://github.com/inveniosoftware/invenio-celery/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

Invenio-Celery could always use more documentation, whether as part of the official Invenio-Celery 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/inveniosoftware/invenio-celery/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 :)

3.1.2 Get Started!

Ready to contribute? Here's how to set up *invenio-celery* for local development.

1. Fork the *inveniosoftware/invenio-celery* repo on GitHub.
2. Clone your fork locally:

   ```
   $ git clone git@github.com:your_name_here/invenio-celery.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 invenio-celery
   $ cd invenio-celery/
   $ pip install -e .[all]
   ```

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 tests:

   ```
   $ ./run-tests.sh
   ```

   The tests will provide you with test coverage and also check PEP8 (code style), PEP257 (documentation), flake8 as well as build the Sphinx documentation and run doctests.

6. Commit your changes and push your branch to GitHub:

   ```
   $ git add .
   $ git commit -s
       -m "component: title without verbs"
       -m "* NEW Adds your new feature."
       -m "* FIX Fixes an existing issue."
       -m "* BETTER Improves and existing feature."
       -m "* Changes something that should not be visible in release notes."
   $ git push origin name-of-your-bugfix-or-feature
   ```

7. Submit a pull request through the GitHub website.

3.1.3 Pull Request Guidelines

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

1. The pull request should include tests and must not decrease test coverage.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring.
3. The pull request should work for Python 2.7, 3.3, 3.4 and 3.5. Check https://travis-ci.org/inveniosoftware/invenio-celery/pull_requests and make sure that the tests pass for all supported Python versions.

3.2 Changes

Version 1.0.1 (released 2018-12-06)

- Adds support for Celery v4.2. Technically this change is backward incompatible because it is no longer possible to load tasks from bare modules (e.g. mymodule.py in the Python root). This is a constraint imposed by Celery v4.2. We however do not known of any cases where bare modules have been used, and also this design is discouraged so we are not flagging it as a backward incompatible change, in order to have the change readily available for current Invenio version.

Version 1.0.0 (released 2018-03-23)

- Initial public release.

3.3 License

MIT License

Copyright (C) 2015-2018 CERN.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Note: In applying this license, CERN does not waive the privileges and immunities granted to it by virtue of its status as an Intergovernmental Organization or submit itself to any jurisdiction.

3.4 Contributors

- Bruno Cuc

- Esteban J. G. Gabancho
- Javier Delgado
- Jiri Kuncar
- Krzysztof Nowak
- Lars Holm Nielsen
- Paulina Lach
- Sami Hiltunen
- Sebastian Witowski
- Tibor Simko

Python Module Index

Index