collective.celery Documentation

Release 1.0a1

Plone Intranet

May 14, 2015

Contents

`collective.celery` provides the necessary bits to use Celery within Plone.

Much of the code here is based off of David Glick's gists, Asko's work and pyramid_celery.

CHAPTER 1

Configuration

Add the python package to your buildout eggs section:

```
eggs =
    ...
# Change this to celery[redis] or celery[librabbitmq] if you want to use Redis or RabbitMQ respective
    celery[sqlalchemy]
    collective.celery
    ...
```

You'll also need to configure buildout to include the celery script in your bin directory:

```
parts =
    ...
    scripts
    ...

[scripts]
recipe = zc.recipe.egg
eggs = ${buildout:eggs}
scripts = pcelery
```

Note: If you already have a `scripts` section, just make sure it also generates pcelery and that the eggs are correct.

Finally, configure celery by setting `environment-vars` on your client configuration. All variables defined there are passed on to celery configuration:

```
environment-vars =
    ...
# CELERY_IMPORTS is required to load your tasks correctly for your project
    CELERY_IMPORTS ('my.package.tasks',)
# basic example just using sqlalchemy
    BROKER_URL sqla+sqlite:///${buildout:directory}/celerydb.sqlite?timeout=30
    CELERY_RESULT_BACKEND db+sqlite:///${buildout:directory}/celeryresults.sqlite?timeout=30
    ...
```

CHAPTER 2

Creating tasks

This package comes with two decorators to use for creating tasks.

default run the task as the user who created the task

as_admin run the task as an admin

Example:

```
from collective.celery import task

@task()
def do_something(context, arg1, foo='bar'):
    pass

@task.as_admin()
def do_something_as_admin(context, arg1, foo='bar'):
    pass
```

And to schedule the taks:

```
my_content_object = self.context
do_something.delay(my_content_object, 'something', foo='bar')
```

Or alternatively:

```
my_content_object = self.context
do_something.apply_async((my_content_object, 'something'), {'foo': 'bar'})
```

Check out calliung tasks in the celery documentation for more details.

Note: You do not need to specify a context object if you don't use it for anything meaningful in the task: the system will already set up the correct site and if you just need that you can obtain it easily (maybe via `plone.api`).

CHAPTER 3

Starting the task runner

The package simply provides a wrapper around the default task runner script which takes an additional zope config parameter:

```
$ bin/pcelery worker parts/instance/etc/zope.conf
```

Note: In order for the worker to start correctly, you should have a ZEO server setup. Else the worker will fail stating it cannot obtain a lock on the database.

CHAPTER 4

Developing and testing

If you are developing, and do not want the hassle of setting up a ZEO server and run the worker, you can set the following in your instance `environment-vars`:

```
environment-vars =
    ...
    CELERY_ALWAYS_EAGER True
# CELERY_IMPORTS is required to load your tasks correctly for your project
    CELERY_IMPORTS ('my.package.tasks',)
# basic example just using sqlalchemy
    BROKER_URL sqla+sqlite:///${buildout:directory}/celerydb.sqlite?timeout=30
    CELERY_RESULT_BACKEND db+sqlite:///${buildout:directory}/celeryresults.sqlite?timeout=30
    ...
```

In this way, thanks to the CELERY_ALWAYS_EAGER setting, celery will not send the task to the worker at all but execute immediately when `delay` or `apply_async` are called.

Similarly, in tests, we provide a layer that does the following:

1. Set `CELERY_ALWAYS_EAGER` for you, so any function you are testing that calls an asyncroinous function will have that function executed after commit (see `execution-model`)
2. Use a simple, in-memory SQLite database to store results

To use it, your package should depend, in its `test` extra requirement, from `collective.celery[test]`:

```
# setup.py
...
setup(name='my.package',
    ...
    extras_require={
        ...
        'test': [
            'collective.celery[test]',
        ],
        ...
    },
...
```

And then, in your `testing.py`:

```
...
from collective.celery.testing import CELERY
...

class MyLayer(PloneSandboxLayer):
```

```
    defaultBases = (PLONE_FIXTURE, CELERY, ...)

...
```

4.1 Complete API and advanced usage

4.1.1 `collective.celery` Package

4.2 Changelog

4.2.1 Changelog

1.0a2 (2015-03-03)

- Initial release

CHAPTER 5

Indices and tables

- genindex
- modindex
- search