sqla-taskq Documentation

Release 0.1

Aurélien Matouillot

June 26, 2015

Contents

sqla-taskq in an asynchronous task queue using SQLAlchemy to store the task. It's minimalist but very useful when you don't want to put in place big system like celery. The supported back end are the same as SQLAlchemy.

CHAPTER 1

Engine settings

```
from sqlalchemy import engine_from_config
import sqla_taskq.models as sqla_taskqm

settings = {
    'sqlalchemy.url': 'sqlite:///tmp/myproject.sqlite'
}
engine = engine_from_config(settings, 'sqlalchemy.')
# You can set the engine for your project here

# sqla-taskq should use the same engine
sqla_taskqm.DBSession.configure(bind=engine)
sqla_taskqm.Base.metadata.bind = engine
```

CHAPTER 2

Creating DB table

sqla-taskq needs only one table named *task*. There are 2 ways to create it:

- You can add this line to your python script which initializes your tables.

```
# The engine should be the same as in your project
models.Base.metadata.create_all(engine)
```

- You can call these commands

```
# Example with a sqlite dialect
export SQLA_TASKQ_SQLALCHEMY_URL=sqlite:////tmp/sqla_taskq.db
sqla_taskq_initializedb

# Your can check you have created the database:
ll /tmp/sqla_taskq.db
```

Note: if a database already exists for the given dialect, the database will not be erased, it will just add the new table.

CHAPTER 3

Inserting a task

You just have to call *Task.create* with the function to call and optionally args and kwargs.

Here is an example:

```
from sqla_taskq.models import Task
args = [1, 2]
kw = {'a': 1, 'b': 2}
Task.create(mymodule.myfunction, args, kw)
```

CHAPTER 4

Running the daemon

Before running the daemon make sure the database is created. There is 3 ways to pass the DB dialect:

- Using the environment variable

```
export SQLA_TASKQ_SQLALCHEMY_URL=sqlite:////tmp/sqla_taskq.db
sqla_taskq_daemon start
```

- Using the '-u' parameters

```
sqla_taskq_daemon start -u sqlite:////tmp/sqla_taskq.db
```

- Using a config file

We suppose we have a file named /tmp/sqla_taskq.ini with this content

```
[sqla_taskq]
sqla_url = sqlite:////tmp/sqla_taskq.db
```

You can call the daemon like this

```
sqla_taskq_daemon start -c /tmp/sqla_taskq.ini

# Daemon status
sqla_taskq_daemon status

# Stop the daemon
sqla_taskq_daemon stop
```

Note: If the daemon is running, passing parameters to the status or the stop function will have not effect.

4.1 Command line and file parameters

Note: You can run sqla_taskq_daemon –help

`-u/--url` <str>: the sqlalchemy url to use (like sqlite:////tmp/sqla_taskq.db) Config file name: `sqla_url`

`-t/--timeoout` <int> (Default: 60s): The timeout in second the daemon wait before killing running task after a stop. No effect if `-k` is passed. Config file name: `timeout`

`-k/--kill`: kill the task when stopping. It will not wait for the end of the current task. Config file name: `kill`

`-c/--config-file` <filename> : Pass a config file to the daemon

Note: The advantage of using config file is to be sure we always use the same conf and to be able to defined logging.

CHAPTER 5

Supervisor

sqla-taskq can be run with supervisor.

You can add this config to your supervisor config or create a new one like in this example file.

```
[program:sqla_taskq]
command=python sqla_taskq/run_supervisor.py
process_name=%(program_name)s-%(process_num)01d
numprocs = 4
```

CHAPTER 6

Demo

You have to clone the repository from github and execute the following command line:

```
git clone https://github.com/LeResKP/sqla-taskq.git
cd sqla-taskq
mkvirtualenv sqla_taskq-env
python setup.py develop
sqla_taskq_initializedb
python sqla_taskq/examples/add_tasks.py
```

Now we will just run the daemon to let it execute the tasks

```
python sqla_taskq/run_daemon.py start
```

You should see the output on your terminal:

```
started with pid XXX
Process started
test_func1 called
test_func1 done
test_func2 called
test_func2 done
```

Now you can stop the daemon

```
python sqla_taskq/run_daemon.py stop
```

6.1 Testing with supervisor

```
python sqla_taskq/examples/add_tasks.py

# Starting supervisor
supervisord -c sqla_taskq/examples/supervisor.conf

# Status
supervisorctl -c sqla_taskq/examples/supervisor.conf status

# Stop the process
supervisorctl -c sqla_taskq/examples/supervisor.conf stop all

# Kill supervisord
killall supervisord
```