Periodic Tasks and Queue Processing in Django

Abhishek Tiwari

Citation: *A. Tiwari*, "Periodic Tasks and Queue Processing in Django", Abhishek Tiwari, 2011. doi:10.59350/cayd4-45249

Published on: September 02, 2011

In Python/Django Celery with RabbitMQ is widely used for background processing or distributed task queue. Although Celery is really focused on being a distributed task queue, it can also be used as scheduler using it's periodic tasks feature celerybeat which kicks off tasks at regular intervals.

Installing

Installing Celery and RabbitMQ is quite easy.

First install RabbitMQ, add a new user and remove guest user. To use celery we need to create a virtual host and allow new user access to that virtual host:

```
sudo apt-get install rabbitmq-server
sudo rabbitmqctl add_user new_user new_password
sudo rabbitmqctl delete_user guest
sudo rabbitmqctl add_vhost myvhost
sudo rabbitmqctl set_permissions -p myvhost new_user ".*" ".*" ".*"
```

If you don't want to use RabbitMQ then you can use Redis, MangoDB or Django Database. For more information see other broker types.

Then install, Celery and django-celery,

```
sudo pip install celery django-celery
```

Using Celery in Djngo

Add `djcelery` to `INSTALLED_APPS`. Then add the following lines to `settings.py`

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

Synchronize the celery database tables::

```
python manage.py syncdb
```

Configure the broker settings, by adding the following to your `settings.py`

```
BROKER_HOST = "localhost"
BROKER_PORT = 5672
BROKER_USER = "new_user"
BROKER_PASSWORD = "new_password"
BROKER_VHOST = "myvhost"
# List of modules to import when celery starts.
CELERY_IMPORTS = ("myapp.tasks", )
```

Use corresponding commands to manage the tasks,

```
python manage.py celeryd
python manage.py celerybeat
python manage.py camqadm
python manage.py celeryev
```

If you are using mod_wsgi in production then you should also add following to .wsgi file,

```
import os
os.environ["CELERY_LOADER"] = "django"
```

Use Cases

###1. djnago-notification

djnago-notification [] in blocking mode can make your application very slow. To queue all notifications add following in settings.py

```
NOTIFICATION_QUEUE_ALL = True
```

Normally you can use,

```
python manage.py emit_signals
```

###2. django-mailer django-mailer is asynchronous.After putting mail on the queue you need to periodically tell it to clear the queue and actually send the mail using,

```
python manage.py send_mail
```

Solution using Celery and RabbitMQ

A better way will be to run Celery tasks in beat mode, so create a `tasks.py` in project root folder (same level to `settings.py`)

```
from celery.task.schedules import crontab
from celery.decorators import periodic_task
# Get django-mailer management function to fire notification queue
from notification.engine import send_all
# Get django-mailer management function to fire mail queue
from mailer.engine import send_all as send_all_mail

# this will run every minute, * is every
@periodic_task(run_every=crontab(hour="*", minute="*", day_of_week="*"))
def project_tasks():
    send_all()
    send_all_mail()
```

Now run the celery daemon `celeryd` in "beat" mode (-B). You can also run celery daemon in background mode with `celeryd_detach`

```
sudo ./manage.py celeryd -v 2 -B -s celery -E -l INFO
sudo ./manage.py celeryd_detach -B -s celery -E -l INFO
```

Running celeryd as a daemon using init script

Although you can use celeryd_detach to run celeryd daemon in background, but in production environment you are better with init scripts as you can start and stop the deamon when you want. Download init script for celeryd and create the file `/etc/init.d/celeryd` add content of downloaded script to it and make it executable.

```
sudo nano /etc/init.d/celeryd
sudo chmod +x /etc/init.d/celeryd
Usage:  /etc/init.d/celeryd {start|stop|restart|status}
```

Now create configuration file: `/etc/default/celeryd` and add following,

```
# Name of nodes to start, here we have a single node
CELERYD_NODES="w1"
# or we could have multiple nodes:
#CELERYD_NODES="w1 w2 w3"

# Where to chdir at start.
#CELERYD_CHDIR="/home/abhi/GIT-DO-NOT-DELETE/biznecto/"
CELERYD_CHDIR="/your/django/project/"

# Name of the projects settings module.
export DJANGO_SETTINGS_MODULE="settings"

# How to call "manage.py celeryd_multi"
CELERYD_MULTI="$CELERYD_CHDIR/manage.py celeryd_multi"

# Extra arguments to celeryd
CELERYD_OPTS="--time-limit=300 --concurrency=8 -B -s celery -E -l INFO"

# Name of the celery config module.
#CELERY_CONFIG_MODULE="celeryconfig"

# %n will be replaced with the nodename. You should create corresponding
    folders
#CELERYD_LOG_FILE="/var/log/celery/%n.log"
#CELERYD_PID_FILE="/var/run/celery/%n.pid"

# Workers should run as an unprivileged user. Avoid running as sudo user.
#CELERYD_USER="celery"
#CELERYD_GROUP="celery"
```