# **longlist\_news**

[**longlist\_news** 1](#_Toc98876473)

[Tasks & Methodology 2](#_Toc98876474)

[Develop a Scheduler to access SerpAPI endpoint for accessing the news items 2](#_Toc98876475)

[Create an endpoint to access serialized data in respect of all tables from the database 2](#_Toc98876476)

[Project 2](#_Toc98876477)

[Database 2](#_Toc98876478)

[Database Tables 3](#_Toc98876479)

[Database Configuration Steps 3](#_Toc98876480)

[Project Setup Steps 5](#_Toc98876481)

[Configure the Project Settings 6](#_Toc98876482)

[Create Admin Superuser 8](#_Toc98876483)

[GIT Steps 8](#_Toc98876484)

[Install Redis on Mac 8](#_Toc98876485)

[Starting the Scheduled Tasks to run with with Celery-Beat 9](#_Toc98876486)

[Terminal window 1: Start the local server 9](#_Toc98876487)

[Terminal window 2: Start the Redis server 9](#_Toc98876488)

[Terminal window 3: Start the Celery Beat Scheduler 10](#_Toc98876489)

[Terminal window 4: Start the worker 11](#_Toc98876490)

[Terminal window 5: May be needed occasionally 12](#_Toc98876491)

[Extra Python Tips 12](#_Toc98876492)

[To retrieve a list of all of the values for one field from a query in django 12](#_Toc98876493)

[To retrieve the related dictionary from a list of dictionaries meeting a search criteria 13](#_Toc98876494)

[Steps for Adding django-crontab 14](#_Toc98876495)

[( If you are using django-crontab for task scheduling, Not used in this project) 14](#_Toc98876496)

[References 16](#_Toc98876497)

[Build a REST API in 30 minutes with Django REST Framework 16](#_Toc98876498)

[Redis on Mac & Linux 16](#_Toc98876499)

[Asynchronous Tasks in Django with Redis and Celery 16](#_Toc98876500)

[Asynchronous Task with Django Celery Redis and Production using Supervisor 16](#_Toc98876501)

[Schedule Tasks in Python Django App with Celery 16](#_Toc98876502)

[How to schedule ‘the Boring Stuff’ with Django and Celery Beat 16](#_Toc98876503)

[Dynamic Task Scheduling With Django-celery-beat 16](#_Toc98876504)

## Tasks & Methodology

### Develop a Scheduler to access SerpAPI endpoint for accessing the news items

The app will search the SerpAPI endpoint at definite intervals for the latest results with each of the keys framed by joining the company names with True status from the Company table with the Keys with True status in the keys table. User will have an option to put the Company Name anywhere he likes in the query string. To obtain this, the key string in the database may contain the **$company** key word in it, that will be replaced by the company name at the time of processing. With each of the items on Page 1 from the resulting JSON dictionary will be checked for

1. The key is named **source**. If the source.name exists in the Postingsite table and if the quality of the Postingsite is True then the JSON dataset will be stored in the Newsitem table if its link does not exist earlier (It should be unique).
2. If the source.name does not exist in the Postingsite table, a new record of Postingsite class is created with True status and the JSON dataset will be stored in the Newsitem table.

The SerpAPI API Key will be kept in .env file and will be accessed by the concerning module at the runtime.

**Using SerpAPI**

[https://serpapi.com/search.json?q=Amazon](https://serpapi.com/search.json?q=Amazone)Hiring&tbm=nws&location=India&api\_key=17c7dedba4e41e0b0e4fb21ea4bafe9267e57615413a812b27bd20ca34b614f6

From the list news\_results the values for the keys link, title, source, snippet and date can be captured from each of the dictionaries on page 1.

### Create an endpoint to access serialized data in respect of all tables from the database

Appropriate views will be developed to meet the requirements.

## Project

Project folder name longlist\_news

Project Name news\_project

App news

Files .gitignore, .env (will contain the API key)

## Database

PostgreSQL Connection Name longlist\_news

Database Name longlist\_news

User longlist

Password longlist

Host 127.0.0.1

Port 5432

## Database Tables

1. **Newsitem** - to store the new results items with the following fields.

title string 255 characters

link string 255 characters

source string 255 characters

snippet string 255 characters

date \_posted date DateType

quality\_source boolean True/False

The field link should be unique.

1. **Key** - to store the search keys

key string

status boolean (True/False) Default True

1. **Postingsite**

name string

quality boolean Default True

1. **Company** to store company names for framing the key to search

name string

status boolean Default True

Verbose\_name\_plural ‘Companies’

## Database Configuration Steps

(1) Check Postgres version

MK:~ purohit$ postgres -V

postgres (PostgreSQL) 13.2

(2)

MK:~ purohit$ which postgres

/usr/local/bin/postgres

(3) Connect to Postgres database

We already have a user longlist (password: longlist) with Create DB Role on Postgres, so connect the database as user longlist.

MK:~ purohit$ psql longlist

psql (13.2)

Type "help" for help.

(4) Check the List of existing users

longlist=# \du

Role name | List of roles : Attributes

---------------------------------------------------------------------------------------------------------------------------------

geo001 | Create DB

longlist | Create DB

postgres | Create role, Create DB, Replication, Bypass RLS

purohit | Superuser, Create role, Create DB, Replication, Bypass RLS

user001 | Create DB

waterwatch | Superuser, Create role, Create DB

(5) Create a new database longlist\_news

longlist=# CREATE DATABASE longlist\_news;

CREATE DATABASE

(6) See the list of databases if longlist\_news has been added to the list

longlist=# \l

Name | Owner |

--------------+---------+----------+---------+-------+------------------------

cptwater | purohit

edsys | purohit

geo | purohit

geodjango | geo001

gis | purohit

longlist | purohit

longlist\_news | purohit

mymap | purohit

nyc | purohit

postgis\_test | purohit

postgres | purohit

template0 | purohit

template1 | purohit

tompkins | purohit

wxstations | purohit

(15 rows)

(7) Connect with the newly created database as user longlist

longlist=# \c longlist\_news longlist

You are now connected to database "longlist\_news" as user "longlist".

(8) Quit the PSQL

longlist\_news=> \q

MK:~ purohit$

## Project Setup Steps

(1)

MK:upwork purohit$ mkdir longlist\_news

MK:upwork purohit$ cd longlist\_news

(2)

Create a file .env having the api\_key entry in it.

Create .gitignore and besides the other entries add *.env* in the list.

Create a folder *resources* and save this document *longlist\_news.doc* in it.

Create a folder *templates*.

(3)

MK:longlist\_news purohit$ git init

Initialized empty Git repository in /Volumes/T7/upwork/longlist\_news/.git/

MK:longlist\_news purohit$ git branch -m main

(4)

MK:longlist\_news purohit$ python3 -m venv env

MK:longlist\_news purohit$ ls

env resources

(5)

MK:longlist\_news purohit$ source env/bin/activate

(6) Install appropriate Python libraries

(env) MK:longlist\_news purohit$ pip install --upgrade pip

Successfully installed pip-22.0.4

(env) MK:longlist\_news purohit$ pip install **django**

Installing collected packages: sqlparse, asgiref, django

Successfully installed asgiref-3.5.0 django-4.0.3 sqlparse-0.4.2

(env) MK:longlist\_news purohit$ pip install **djangorestframework**

Successfully installed djangorestframework-3.13.1 pytz-2021.3

(env) MK:longlist\_news purohit$ pip install **psycopg2-binary**

Successfully installed psycopg2-binary-2.9.3

(env) MK:longlist\_news purohit$ pip install **python-dateutil**

Successfully installed python-dateutil-2.8.2 six-1.16.0

( If you are using django-crontab for task scheduling - Not used for this project)

(env) MK:longlist\_news purohit$ pip install **django-crontab**

Successfully installed django-crontab-0.7.1

(env) MK:longlist\_news purohit$ pip install **python-decouple**

Successfully installed python-decouple-3.6

(env) MK:longlist\_news purohit$ pip install **celery**

Successfully installed amqp-5.1.0 billiard-3.6.4.0 celery-5.2.3 click-8.0.4 click-didyoumean-0.3.0 click-plugins-1.1.1 click-repl-0.2.0 kombu-5.2.4 prompt-toolkit-3.0.28 setuptools-59.6.0 vine-5.0.0 wcwidth-0.2.5

(env) MK:longlist\_news purohit$ pip install **redis**

Successfully installed deprecated-1.2.13 packaging-21.3 pyparsing-3.0.7 redis-4.1.4 wrapt-1.14.0

(env) MK:longlist\_news purohit$ pip install **django-celery-beat**

Successfully installed Django-3.2.12 django-celery-beat-2.2.1 django-timezone-field-4.2.3 python-crontab-2.6.0

*(Observe that Django-4.0.3 was installed earlier but that was uninstalled during the installation process of djano-celery-beat and an older version of Django-3.2.12 was installed.)*

(env) MK:longlist\_news purohit$ django-admin startproject news\_project .

(env) MK:longlist\_news purohit$ python manage.py startapp news

(env) MK:longlist\_news purohit$ python manage.py makemigrations

(env) MK:longlist\_news purohit$ python manage.py migrate

## Configure the Project Settings

***edit news\_project/settings.py : (added or edited text in italics)***

INSTALLED\_APPS = [

*'longlist\_app.apps.LonglistAppConfig',*

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

*'rest\_framework',*

*'django\_celery\_beat'*

]

DATABASES = {

'default': {

'ENGINE': *'django.db.backends.postgresql',*

'NAME': *'longlist\_news'*,

'USER': *'longlist',*

'PASSWORD': 'longlist',

'HOST': '127.0.0.1',

'PORT': '5432',

}

}

TEMPLATES = [

{

'BACKEND': 'django.template.backends.django.DjangoTemplates',

'DIRS': [*BASE\_DIR / 'templates'*],

*# BASE\_DIR is the project path, that contains the complete project including project\_site folder (longlist\_site), env, templates and project\_app (longlist\_app) folders.*

*………………*

*………………*

]

TIME\_ZONE = *'Asia/Kolkata'*

*# Celery*

*CELERY\_BROKER\_URL = 'redis://localhost:6379'*

*CELERY\_RESULT\_BACKEND = 'redis://localhost:6379'*

*CELERY\_ACCEPT\_CONTENT = ['application/json']*

*CELERY\_TASK\_SERIALIZER = 'json'*

*CELERY\_RESULT\_SERIALIZER = 'json'*

*CELERY\_BEAT\_SCHEDULER = 'django\_celery\_beat.schedulers:DatabaseScheduler'*

***edit news\_project/\_\_init\_\_.py : (add the following text, in italics)***

*from .celery import celery\_app*

*\_\_all\_\_ = ('celery\_app',)*

***Add news\_project/celery.py : (added text in italics)***

*import os*

*from celery import Celery*

*# Set default Django settings*

*os.environ.setdefault('DJANGO\_SETTINGS\_MODULE', 'news\_project.settings')*

*celery\_app = Celery('news\_project')*

*# Celery will apply all configuration keys with defined namespace*

*celery\_app.config\_from\_object('django.conf:settings', namespace = 'CELERY')*

*# Load tasks from all registered apps*

*celery\_app.autodiscover\_tasks()*

## Create Admin Superuser

(env) MK:longlist\_news purohit$ python manage.py createsuperuser

Username (leave blank to use 'purohit'): purohit

Email address: mkpurohit.imd2007@gmail.com

Password: longlist\_news

Password (again): longlist\_news

Superuser created successfully.

(env) MK:longlist\_news purohit$

## GIT Steps

**Create a new repository on the command line**

git init

git status

git add -A

git status

git commit -m "first commit"

git branch -M master

git remote add origin https://github.com/mk-purohit/longlist\_news.git

git push -u origin master

**Push an existing repository from the command line**

git branch -M master

git push -u origin master

## Install Redis on Mac

* Install [Homebrew](https://brew.sh/):

**/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install.sh)"**

* Install redis

**brew install redis**

Every once and a while run **brew upgrade redis**

* Start / Stop Redis

**brew services start redis**

* Starting redis this way turns redis into a background service. You can easily stop redis with:

**brew services stop redis**

* Verify redis is running:

**redis-cli ping**

What result do you see?

* + PONG -- great, redis is working and ready.
  + Could not connect to Redis at 127.0.0.1:6379: Connection refused -- this means that: (1) you did not install redis correctly or (2) redis is not running.

## Starting the Scheduled Tasks to run with with Celery-Beat

### Terminal window 1: Start the local server

**(env) MK:longlist\_news purohit$ python manage.py runserver**

Watching for file changes with StatReloader

Performing system checks...

System check identified no issues (0 silenced).

March 21, 2022 - 15:38:15

Django version 3.2.12, using settings 'news\_project.settings'

Starting development server at http://127.0.0.1:8000/

Quit the server with CONTROL-C.

### Terminal window 2: Start the Redis server

**MK:~ purohit$ redis-server**

You should see the following on the terminal

…………………………..

…………………………..

\_.\_

\_.-``\_\_ ''-.\_

\_.-`` `. `\_. ''-.\_ Redis 6.2.6 (00000000/0) 64 bit

.-`` .-```. ```\/ \_.,\_ ''-.\_

( ' , .-` | `, ) Running in standalone mode

|`-.\_`-...-` \_\_...-.``-.\_|'` \_.-'| Port: 6379

| `-.\_ `.\_ / \_.-' | PID: 45894

`-.\_ `-.\_ `-./ \_.-' \_.-'

|`-.\_`-.\_ `-.\_\_.-' \_.-'\_.-'|

| `-.\_`-.\_ \_.-'\_.-' | https://redis.io

`-.\_ `-.\_`-.\_\_.-'\_.-' \_.-'

|`-.\_`-.\_ `-.\_\_.-' \_.-'\_.-'|

| `-.\_`-.\_ \_.-'\_.-' |

`-.\_ `-.\_`-.\_\_.-'\_.-' \_.-'

`-.\_ `-.\_\_.-' \_.-'

`-.\_ \_.-'

`-.\_\_.-'

…………………………..

…………………………..

\* Ready to accept connections

If you see the following error:

………………

………………

Failed listening on port 6379 (TCP), aborting

**killall redis-server** and re-run **redis-server**

### Terminal window 3: Start the Celery Beat Scheduler

**(env) MK:longlist\_news purohit$ celery -A news\_project beat -l INFO --scheduler django\_celery\_beat.schedulers:DatabaseScheduler**

celery beat v5.2.3 (dawn-chorus) is starting.

\_\_ - ... \_\_ - \_

LocalTime -> 2022-03-21 16:56:54

Configuration ->

. broker -> redis://localhost:6379//

. loader -> celery.loaders.app.AppLoader

. scheduler -> django\_celery\_beat.schedulers.DatabaseScheduler

. logfile -> [stderr]@%INFO

. maxinterval -> 5.00 seconds (5s)

[2022-03-21 16:56:54,673: INFO/MainProcess] beat: Starting...

[2022-03-21 16:56:54,887: INFO/MainProcess] Scheduler: Sending due task Hello World (news.tasks.hello\_world)

[2022-03-21 16:56:54,976: INFO/MainProcess] Scheduler: Sending due task create file (news.tasks.my\_first\_task)

### Terminal window 4: Start the worker

**(env) MK:longlist\_news purohit$ celery -A news\_project worker -l INFO**

-------------- celery@MK.local v5.2.3 (dawn-chorus)

--- \*\*\*\*\* -----

-- \*\*\*\*\*\*\* ---- macOS-12.2.1-x86\_64-i386-64bit 2022-03-21 16:58:12

- \*\*\* --- \* ---

- \*\* ---------- [config]

- \*\* ---------- .> app: news\_project:0x110dba580

- \*\* ---------- .> transport: redis://localhost:6379//

- \*\* ---------- .> results: redis://localhost:6379/

- \*\*\* --- \* --- .> concurrency: 4 (prefork)

-- \*\*\*\*\*\*\* ---- .> task events: OFF (enable -E to monitor tasks in this worker)

--- \*\*\*\*\* -----

-------------- [queues]

.> celery exchange=celery(direct) key=celery

[tasks]

. news.tasks.hello\_world

. news.tasks.my\_first\_task

[2022-03-21 16:58:12,497: INFO/MainProcess] Connected to redis://localhost:6379//

[2022-03-21 16:58:12,505: INFO/MainProcess] mingle: searching for neighbors

[2022-03-21 16:58:13,517: INFO/MainProcess] mingle: all alone

[2022-03-21 16:58:13,542: WARNING/MainProcess] /Volumes/T7/upwork/longlist\_news/env/lib/python3.9/site-packages/celery/fixups/django.py:203: UserWarning: Using settings.DEBUG leads to a memory leak, never use this setting in production environments!

warnings.warn('''Using settings.DEBUG leads to a memory

[2022-03-21 16:58:13,542: INFO/MainProcess] celery@MK.local ready.

[2022-03-21 21:02:55,474:INFO/MainProcess] Task news.tasks.my\_first\_task[5bb3a379-9833-4199-864f-0a227e70a137] received

[2022-03-21 21:02:55,491: INFO/ForkPoolWorker-2] Task news.tasks.my\_first\_task[5bb3a379-9833-4199-864f-0a227e70a137] succeeded in 0.009380346000398276s: None

[2022-03-21 21:03:25,486: INFO/MainProcess] Task news.tasks.hello\_world[65819da0-e6d1-4ddf-b68a-04a40e31e131] received

[2022-03-21 21:03:25,488: WARNING/ForkPoolWorker-2] Hello world!

[2022-03-21 21:03:25,490: INFO/ForkPoolWorker-2] Task news.tasks.hello\_world[65819da0-e6d1-4ddf-b68a-04a40e31e131] succeeded in 0.0024534410003980156s: None

### Terminal window 5: May be needed occasionally

Sometimes the tasks do not run as per the schedule, particularly, when you add a new task.

To fix that you would have to reset the “last run time” for each periodic task:

In a separate terminal window, activate the environment and run the python shell and the following script.

Python manage.py shell

>>> from django\_celery\_beat.models import PeriodicTask, PeriodicTasks

>>> PeriodicTask.objects.all().update(last\_run\_at=None)

>>> for task in PeriodicTask.objects.all():

>>> PeriodicTasks.changed(task)

**Note**

This will reset the state as if the periodic tasks have never run before.

## Extra Python Tips

### To retrieve a list of all of the values for one field from a query in django

For example, I have a query of users, but rather than a queryset (or list) of user objects, I want a list just the usernames (strings). In a sense this is asking to restrict only to one column of data.

list(User.objects.all().values\_list('username', flat=True))

If you only pass in a single field, you can also pass in the flat parameter. If True, this will mean the returned results are single values, rather than one-tuples. Additionally, casting it to a list makes the returned value a list instead of a queryset.

list(Postingsite.objects.all().values\_list('name’', ‘quality’))

If you pass more than one field, you can not pass the flat parameter. In that case, you will be returned with a list of tuples, each tuple having the values for each field passed.

list(Postingsite.objects.all().values('name’'))

A list of dictionaries with the one key, name will be returned, If you pass only one field with **values** and not **values\_list** option. In this case also, you can not pass the flat parameter.

list(Postingsite.objects.all().values('name’', ‘quality’))

A list of dictionaries with the two keys, name and quality, will be returned, If you pass more than one field with **values** and not **values\_list** option. you can not pass the flat parameter.

In all the above cases, if you don’t wrap the query within **list**, you will get a queryset object with lists or dictionaries whatever the case may be like,

(Postingsite.objects.all().values('name’', ‘quality’))

will return a queryset object having a list of dictionaries.

### To retrieve the related dictionary from a list of dictionaries meeting a search criteria

Assume I have this:

[

{"name": "Tom", "age": 10},

{"name": "Mark", "age": 5},

{"name": "Pam", "age": 7}

]

and by searching "Pam" as name, I want to retrieve the related dictionary: {name: "Pam", age: 7}

How to achieve this ?

Use a [generator expression](http://www.python.org/dev/peps/pep-0289/):

>>> dicts = [

... { "name": "Tom", "age": 10 },

... { "name": "Mark", "age": 5 },

... { "name": "Pam", "age": 7 },

... { "name": "Dick", "age": 12 }

... ]

>>> next(item for item in dicts if item["name"] == "Pam")

{'age': 7, 'name': 'Pam'}

If you need to handle the item not being there, then you can do what user [Matt](https://stackoverflow.com/users/277250/matt) [suggested in his comment](https://stackoverflow.com/questions/8653516/python-list-of-dictionaries-search#comment18634157_8653568) and provide a default using a slightly different API:

next((item for item in dicts if item["name"] == "Pam"), None)

And to find the index of the item, rather than the item itself, you can [enumerate()](https://docs.python.org/3/library/functions.html#enumerate) the list:

next((i for i, item in enumerate(dicts) if item["name"] == "Pam"), None)

**A more pythonic way to the problem:**

people = [

{'name': "Tom", 'age': 10},

{'name': "Mark", 'age': 5},

{'name': "Pam", 'age': 7}

]

filter(lambda person: person['name'] == 'Pam', people)

result (returned as a list in Python 2):

[{'age': 7, 'name': 'Pam'}]

Note: In Python 3, a filter object is returned. So the python3 solution would be:

list(filter(lambda person: person['name'] == 'Pam', people))

## Steps for Adding django-crontab

## ( If you are using django-crontab for task scheduling, Not used in this project)

refer<https://pypi.org/project/django-crontab/>

1) pip install django-crontab

2) Add it to installed apps in django settings.py:

INSTALLED\_APPS = (

'django\_crontab',

...,

)

3) Create a new method (define a function) that should be executed by cron let at every 15th minute past every hour from 9 to 23 Hrs., e.g. *rains.views.process\_gridded\_rains,* Here, *process\_gridded\_rains* is a function, defined in *views.py* file in *rains* app, like,

def process\_gridded\_rains():

…….

……..

Return

4) Add this to your settings.py:

CRONJOBS = [

('\*/15 9-23 \* \* \*', 'rains.views.process\_gridded\_rains','

>> /Volumes/T7/projects/edsys/logs/rains\_gridded\_job.log'),

]

The syntax of Cron jobs is \* \* \* \* \*, explained in the following table.

Refer<https://crontab.guru/>

| **Description** | **\* (1st star)**  **Minute** | **\* (2nd star)**  **Hour** | **\* (3rd star)**  **Day (Month)** | **\* (4th star)**  **Month** | **\* (5th star)**  **Day (Week)** |
| --- | --- | --- | --- | --- | --- |
| **Any Value** | **\*** | **\*** | **\*** | **\*** | **\*** |
| **Value List Separator** | **,** | **,** | **,** | **,** | **,** |
| **Range of Values** | **-** | **-** | **-** | **-** | **-** |
| **Step Values** | **/** | **/** | **/** | **/** | **/** |
| **Allowed Values** | **0-59** | **0-23** | **1-31** | **1-12** | **0-6** |
| **Alternative Single Values** |  |  |  | **JAN-DEC** | **SUN-SAT** |
| **Sunday (Non-standard)** |  |  |  |  | **7** |

5) Run this command to add all defined jobs from *CRONJOBS* to crontab (of the user which you are running this command with):

python manage.py crontab add

6) Show current active jobs of this project:

python manage.py crontab show

7) Removing all defined jobs is straight forward:

python manage.py crontab remove

8) NOTE: Run “python manage.py crontab add” each time you change CRONJOBS in any way!

9) examples:

CRONJOBS = [

('\*/5 \* \* \* \*', 'myapp.cron.my\_scheduled\_job'),

# format 1

('0 0 1 \* \*', 'myapp.cron.my\_scheduled\_job', '>> /tmp/scheduled\_job.log'),

# format 2

('0 0 1 \* \*', 'myapp.cron.other\_scheduled\_job', ['myapp']),

('0 0 \* \* 0', 'django.core.management.call\_command', ['dumpdata', 'auth'], {'indent': 4}, '> /home/john/backups/last\_sunday\_auth\_backup.json'),

]

## References

### Build a REST API in 30 minutes with Django REST Framework

<https://medium.com/swlh/build-your-first-rest-api-with-django-rest-framework-e394e39a482c>

### 

### Redis on Mac & Linux

<https://www.codingforentrepreneurs.com/blog/install-redis-mac-and-linux>

### 

### Asynchronous Tasks in Django with Redis and Celery

<https://stackabuse.com/asynchronous-tasks-in-django-with-redis-and-celery/>

### Asynchronous Task with Django Celery Redis and Production using Supervisor

<https://medium.com/swlh/asynchronous-task-with-django-celery-redis-and-production-using-supervisor-ef920725da03>

### Schedule Tasks in Python Django App with Celery

<https://python.plainenglish.io/schedule-tasks-in-python-django-app-with-celery-692b6e626230>

### How to schedule ‘the Boring Stuff’ with Django and Celery Beat

<https://www.merixstudio.com/blog/django-celery-beat/>

### Dynamic Task Scheduling With Django-celery-beat

<https://medium.com/swlh/dynamic-task-scheduling-with-django-celery-beat-f2591d52e15>