django-q-registry

Josh Thomas

Mar 23, 2024

DEVELOPMENT

A Django app to register periodic Django Q tasks.

CHAPTER

ONE

REQUIREMENTS

- Python 3.8, 3.9, 3.10, 3.11, 3.12
- Django 3.2, 4.2, 5.0
- Django Q2 1.4.3+
  - This package has only been tested with the Django ORM broker.

CHAPTER

TWO

GETTING STARTED

1. Install the package from PyPI:

```
python -m pip install django-q-registry
```

2. Add the app to your Django project's `INSTALLED_APPS`:

```
INSTALLED_APPS = [
    ...,
    "django_q_registry",
    ...,
]
```

CHAPTER

THREE

USAGE

3.1 Registering Periodic Tasks

There are three supported ways to register periodic tasks:

1. In a `tasks.py` file in a Django app, using the `@register_task` decorator:

```
# tasks.py
from django.core.mail import send_mail
from django_q.models import Schedule
from django_q_registry import register_task


@register_task(
    name="Send periodic test email",
    schedule_type=Schedule.CRON,
    # https://crontab.guru/#*/5_*_*_*_*
    cron="*/5 * * * *",
)
def send_test_email():
    send_mail(
        subject="Test email",
        message="This is a test email.",
        from_email="noreply@example.com",
        recipient_list=["johndoe@example.com"],
    )
```

2. In a `tasks.py` file in a Django app, using the `registry.register` function directly:

```
# tasks.py
from django.core.mail import send_mail
from django_q.models import Schedule
from django_q_registry.registry import registry


registry.register(
    send_mail,
    name="Send periodic test email",
    kwargs={
        "subject": "Test email",
        "message": "This is a test email.",
```

(continues on next page)

(continued from previous page)

```
        "from_email": "noreply@example.com",
        "recipient_list": ["janedoe@example.com"],
    },
    schedule_type=Schedule.CRON,
    # https://crontab.guru/#*/5_*_*_*_*
    cron="*/5 * * * *",
)
```

3. In a Django project's `settings.py` file, using the `Q_REGISTRY["TASKS"]` setting:

```
# settings.py
from django_q.models import Schedule


Q_REGISTRY = {
    "TASKS": [
        {
            "name": "Send periodic test email",
            "func": "django.core.mail.send_mail",
            "kwargs": {
                "subject": "Test email",
                "message": "This is a test email.",
                "from_email": "noreply@example.com",
                "recipient_list": ["janedoe@example.com"],
            },
            "schedule_type": Schedule.CRON,
            # https://crontab.guru/#*/5_*_*_*_*
            "cron": "*/5 * * * *",
        },
    ],
}
```

3.2 Setting up Periodic Tasks in Production

At some point in your project's deployment process, run the `setup_periodic_tasks` management command:

```
python manage.py migrate
python manage.py setup_periodic_tasks
```

This command automatically registers periodic tasks from `tasks.py` files in Django apps, and from the `Q_REGISTRY["TASKS"]` setting. It also cleans up any periodic tasks that are no longer registered.

CHAPTER

FOUR

DOCUMENTATION

Please refer to the documentation for more information.

CHAPTER

FIVE

LICENSE

`django-q-registry` is licensed under the MIT license. See the *`LICENSE`* file for more information.

5.1 Justfile

This project uses Just as a command runner.

The following commands are available:

- *bootstrap*
- *copier-copy*
- *copier-update*
- *copier-update-all*
- *coverage*
- *docs-build*
- *docs-install*
- *docs-serve*
- *fmt*
- *lint*
- *makemigrations*
- *manage*
- *migrate*
- *pup*
- *test*
- *testall*
- *types*

5.1.1 Commands

```
$ just --list
```

```
Available recipes:
    bootstrap
    copier-copy TEMPLATE_PATH DESTINATION_PATH="." # apply a copier template to project
    copier-update ANSWERS_FILE *ARGS # update the project using a copier answers file
    copier-update-all *ARGS # loop through all answers files and update the project␣
↪using copier
    coverage
    docs-build LOCATION="docs/_build/html"
    docs-install
    docs-serve
    fmt                     # format justfile
    lint                    # run pre-commit on all files
    makemigrations *APPS
    mm *APPS                # alias for `makemigrations`
    manage *COMMAND
    migrate *ARGS
    pup
    test *ARGS
    testall *ARGS
    types
```

bootstrap

```
$ just bootstrap
```

```
bootstrap:
    @just pup
    python -m uv pip install --editable '.[dev]'
```

copier-copy

```
$ just copier-copy
```

```
# apply a copier template to project
copier-copy TEMPLATE_PATH DESTINATION_PATH=".":
    copier copy {{ TEMPLATE_PATH }} {{ DESTINATION_PATH }}
```

copier-update

```
$ just copier-update
```

```
# update the project using a copier answers file
copier-update ANSWERS_FILE *ARGS:
    copier update --trust --answers-file {{ ANSWERS_FILE }} {{ ARGS }}
```

copier-update-all

```
$ just copier-update-all
```

```
# loop through all answers files and update the project using copier
@copier-update-all *ARGS:
    for file in `ls .copier/`; do just copier-update .copier/$file "{{ ARGS }}"; done
```

coverage

```
$ just coverage
```

```
coverage:
    python -m nox --session "coverage"
```

docs-build

```
$ just docs-build
```

```
@docs-build LOCATION="docs/_build/html":
    just _cog
    sphinx-build docs {{ LOCATION }}
```

docs-install

```
$ just docs-install
```

```
@docs-install:
    @just pup
    python -m uv pip install 'django-q-registry[docs] @ .'
```

docs-serve

```
$ just docs-serve
```

```
@docs-serve:
    #!/usr/bin/env sh
    just _cog
    if [ -f "/.dockerenv" ]; then
        sphinx-autobuild docs docs/_build/html --host "0.0.0.0"
    else
        sphinx-autobuild docs docs/_build/html --host "localhost"
    fi
```

fmt

```
$ just fmt
```

```
# format justfile
fmt:
    just --fmt --unstable
```

lint

```
$ just lint
```

```
# run pre-commit on all files
lint:
    python -m nox --session "lint"
```

makemigrations

```
$ just makemigrations
```

```
makemigrations *APPS:
    @just manage makemigrations {{ APPS }}
```

manage

```
$ just manage
```

```
manage *COMMAND:
    #!/usr/bin/env python
    import sys

    try:
```

(continues on next page)

(continued from previous page)

```
        from django.conf import settings
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc

    settings.configure(
        INSTALLED_APPS=[
            "django.contrib.contenttypes",
            "django_q",
            "django_q_registry"
        ],
        SECRET_KEY="not needed",
    )
    execute_from_command_line(sys.argv + "{{ COMMAND }}".split(" "))
```

migrate

```
$ just migrate
```

```
migrate *ARGS:
    @just manage migrate {{ ARGS }}
```

pup

```
$ just pup
```

```
pup:
    python -m pip install --upgrade pip uv
```

test

```
$ just test
```

```
test *ARGS:
    python -m nox --session "test" -- "{{ ARGS }}"
```

testall

```
$ just testall
```

```
testall *ARGS:
    python -m nox --session "tests" -- "{{ ARGS }}"
```

types

```
$ just types
```

```
types:
    python -m nox --session "mypy"
```