Django Simple Task

Jun 24, 2021

Contents

`django-simple-task` runs background tasks in Django 3 without requiring other services and workers. It runs them in the same event loop as your ASGI application. It is not resilient as a proper task runner such as Celery, but works for some simple tasks and has less overall overheads.

CHAPTER 1

Requirements

`django-simple-task` expect ASGI lifespan protocol to be supported by the server. Currently Daphne does not support this.

This package is tested with:

- Python 3.7, 3.8 and 3.9,
- Django 3.1 and 3.2.

CHAPTER 2

Guide

Install the package:

```
pip install django-simple-task
```

Added it to installed apps:

```
# settings.py
INSTALLED_APPS = [
    ...
    'django_simple_task'
]
```

Apply ASGI middleware :

```
# asgi.py
from django_simple_task import django_simple_task_middlware
application = django_simple_task_middlware(application)
```

Call a background task in Django view:

```
from django_simple_task import defer

def task1():
    time.sleep(1)
    print("task1 done")

async def task2():
    await asyncio.sleep(1)
    print("task2 done")

def view(requests):
    defer(task1)
    defer(task2)
    return HttpResponse(b"My View")
```

It is required to run Django with ASGI server. Official Doc

CHAPTER 3

Configurations

Concurrency level can be controlled by adding `DJANGO_SIMPLE_TASK_WORKERS` to settings. Defaults to `1`.

CHAPTER 4

Table of Content

4.1 How It Works

Here's a simple overview of how it works:

1. On application start, a queue is created and a number of workers starts to listen to the queue
2. When `defer` is called, a task(function or coroutine function) is added to the queue
3. When a worker gets a task, it runs it or delegates it to a threadpool
4. On application shutdown, it waits for tasks to finish before exiting ASGI server

On a more detailed note, `django_simple_task_middlware` hooks up to ASGI lifespan calls, and on application start up it creates a `asyncio.Queue` and save it as well as the running event loop in AppConfig. A number of workers (defaults to 1) would then start to listen to the queue. When `django_simple_task.defer` is called, it gets the event loop and queue from `AppConfig` and put the task in the queue by calling `call_soon_threadsafe` on the loop.

A queue is used here so that we can have more control over concurrency level, and it would be helpful if we want to add more complex things such as retry mechanism. More importantly, on application shutdown, it allows us to make sure the task queues are finished before exiting ASGI server.

4.2 APIs

4.2.1 django_simple_task

`django_simple_task.`**`defer`**(*func: Union[Callable[[Any], Awaitable[T_co]], Callable], arguments: Optional[Dict[KT, VT]] = None, *, options: Optional[Dict[KT, VT]] = None*)

Adds a function or coroutine function to the task queue

Parameters

- **`func`** – function or coroutine function to be enqueued

- **`arguments`** – optional. In the format of {“args”: [], “kwargs”: {}}
- **`options`** – optional. In the format of { “thread_sensitive”: bool, this gets passed to asgiref.sync.sync_to_async, defaults to False.}

`django_simple_task.`**`django_simple_task_middlware`**(*app*, *, *asgi_version=3*)

Python Module Index

Index