docker-compose + Django + postgreSQL + nginxで開発環境を構築

Django

はじめに

初めにdocker-compose + django + postgreSQLのシンプルな環境を作り、その後docker-compose + django + postgreSQL + nginxでより実践に近い環境を作ろうと思います。

dockerやdocker-composeはすでにインストールされている前提で進めていきます。

環境はこんな感じ。

- macOS Mojave 10.14.5
- Docker: 18.06.1-ce-mac73
- Python: 3.7
- Django: 2.2.2
- PostgreSQL: 11
- psycopg: 2.8.3
- uWSGI: 2.0.18

開発環境の構築（docker-compose + django + postgreSQL）

dockerの公式ドキュメントを参考に進めていきます。

後のnginxの利用も考えて、最終的に以下のようなディレクトリ構成になるようにします。

```
.
├── docker-compose.yml
├── src
│   ├── manage.py
```

```
    └── mysite
        ├── __init__.py
        ├── settings.py
        ├── urls.py
        └── wsgi.py
└── web
    ├── Dockerfile
    └── requirements.txt
```

Dockerfile

バージョンの指定以外は公式ドキュメントの通りです。

```
FROM python:3.7
ENV PYTHONUNBUFFERED 1
RUN mkdir /code
WORKDIR /code
COPY requirements.txt /code/
RUN pip install -r requirements.txt
COPY . /code/
```

requirements.txt

`pip` でインストールするものをここに記述します。

バージョンはこのブログの投稿時点での最新のものです。

```
Django==2.2.2
psycopg2==2.8.3
```

docker-compose.yml

PostgreSQLのバージョンは11に指定しています。

また、コンテナごとにディレクトリを分けているので、webコンテナの `build` と `volumes` のパスが公式ドキュメントとは異なる点に注意してください。

```
version: '3'

services:
  db:
    image: postgres:11
  web:
    build: ./web
    command: python manage.py runserver 0.0.0.0:8000
    volumes:
      - ./src:/code
    ports:
```

```
      - "8000:8000"
    depends_on:
      - db
```

プロジェクトの作成

以下のコマンドを使って、docker上でdjangoのプロジェクトを作成します。

`run` コマンドは指定のサービス（今ならweb）のコンテナ内でコマンドを実行するものです。

```
$ docker-compose run web django-admin startproject mysite .
```

コマンドが正常に終了すると、ローカルに `src/mysite/settings.py` が作成されるので、それを以下のように編集します。

これはdjangoがデータベースとしてPostgreSQLを利用するための設定です。

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'postgres',
        'USER': 'postgres',
        'PASSWORD': 'password',
        'HOST': 'db',
        'PORT': 5432,
    }
}
```

動作確認

コンテナを立ち上げて、ブラウザでページが正しく表示されるか確認します。

`docker-compose.yml` があるディレクトリで以下のコマンドを実行してください。

```
$ docker-compose up
```

ブラウザで `localhost:8000` にアクセスして、以下の画面が表示されれば成功です。

The install worked successfully! Congratulations!

You are seeing this page because DEBUG=True is in your settings file and you have not configured any URLs.

開発環境の構築（docker-compose + django + nginx）

続いて、docker-compose + django + nginxでの開発環境の構築方法についてです。

先ほど作成した環境にnginxの設定を追加する形で進めていきます。

最終的には以下のようなディレクトリ構成になります。

```
.
├── docker-compose.yml
├── nginx
│   ├── conf
│   │   └── mysite_nginx.conf
│   └── uwsgi_params
├── src
│   ├── manage.py
│   ├── mysite
│   │   ├── __init__.py
│   │   ├── __pycache__
│   │   │   ├── __init__.cpython-37.pyc
│   │   │   ├── settings.cpython-37.pyc
│   │   │   ├── urls.cpython-37.pyc
│   │   │   └── wsgi.cpython-37.pyc
│   │   ├── settings.py
│   │   ├── urls.py
│   │   └── wsgi.py
│   └── static
└── web
    ├── Dockerfile
    └── requirements.txt
```

uWSGIの導入

nginxをサーバとして利用する場合、クライアントはDjangoではなくnginxにリクエストを投げることになります。

nginxはリクエスト内容を見て、レスポンスを返したり、djangoにリクエストを渡したりします。

しかし、nginxとdjangoは直接通信できないので、2つをつなぐuWSGIというWSGI (Web Server Gateway Interface) 必要です。

uWSGIは `pip` でインストールできるので、 `requirements.txt` を以下のように記述します。

```
Django==2.2.2
psycopg2==2.8.3
uwsgi==2.0.18
```

uWSGIには設定ファイルである `uwsgi_params` が必要です。

以下をコピペして使ってください。

中身は気にする必要はありません。

```
uwsgi_param  QUERY_STRING       $query_string;
uwsgi_param  REQUEST_METHOD     $request_method;
uwsgi_param  CONTENT_TYPE       $content_type;
uwsgi_param  CONTENT_LENGTH     $content_length;

uwsgi_param  REQUEST_URI        $request_uri;
uwsgi_param  PATH_INFO          $document_uri;
uwsgi_param  DOCUMENT_ROOT      $document_root;
uwsgi_param  SERVER_PROTOCOL    $server_protocol;
uwsgi_param  REQUEST_SCHEME     $scheme;
uwsgi_param  HTTPS              $https if_not_empty;

uwsgi_param  REMOTE_ADDR        $remote_addr;
uwsgi_param  REMOTE_PORT        $remote_port;
uwsgi_param  SERVER_PORT        $server_port;
uwsgi_param  SERVER_NAME        $server_name;
```

nginxコンテナの作成と設定

nginxのコンテナを以下のように定義します。

8000番ポートを開けて、設定ファイルや `./static/` ディレクトリをマウントしています。

`./static/` のマウントはcss等の静的ファイルをnginxから配信するためです。

```
nginx:
    image: nginx:alpine
```

```
    container_name: nginx
    ports:
      - "8000:8000"
    volumes:
      - ./nginx/conf:/etc/nginx/conf.d
      - ./nginx/uwsgi_params:/etc/nginx/uwsgi_params
      - ./src/static:/static
    depends_on:
      - web
```

nginxのコンテナからdjangoのコンテナにリクエストを流すために、設定ファイル `mysite_nginx.conf` を追加します。

ローカルの8000番ポートでリクエストを受けて、 `/static` なリクエストであればnginxがレスポンスを返し、それ以外はdjangoにリクエストを流すという様な設定です。

```
upstream django {
    ip_hash;
    server web:8001;
}

server {
    listen      8000;
    server_name 127.0.0.1;
    charset     utf-8;

    client_max_body_size 75M;

    location /static {
        alias /static;
    }

    location / {
        uwsgi_pass  django;
        include     /etc/nginx/uwsgi_params;
    }
}
```

nginxおよびuWSGIの設定に関しては、以下のサイトが分かりやすいです。

uwsgi-docs.readthedocs.io

www.mathpython.com

www.python.ambitious-engineer.com

その他のコンテナの設定

nginxコンテナ追加後のdocker-compose.ymlは以下のようになります。

説明はコメントで入れています。

```
version: '3'

# データベースのデータを永続化する
# 作成されたvolumeは docker volume ls で確認できる
volumes:
  dbdata:

services:
  nginx:
    image: nginx:alpine
    container_name: nginx
    ports:
      - "8000:8000"
    volumes:
      - ./nginx/conf:/etc/nginx/conf.d
      - ./nginx/uwsgi_params:/etc/nginx/uwsgi_params
      - ./src/static:/static
    depends_on:
      - web

  db:
    image: postgres:11
    container_name: db
    ports:
      - "5432:5432"
     # トップレベルで指定したvolumeをマウントする
    volumes:
      - "dbdata:/var/lib/postgresql/data"
    environment:
      POSTGRES_PASSWORD: password # settings.pyで設定した値と同じにする

  web:
    build: ./web
    container_name: web
     # uwsgiを使ってアプリケーションを起動させる
    command: uwsgi --socket :8001 --module mysite.wsgi
    volumes:
      - ./src:/code
      - ./src/static:/static
    expose:
      - "8001"
    depends_on:
      - db
```

staticディレクトリの設定

ページにcssやjavascript等の静的ファイルを反映させるために、`src/mysite/settings.py`に以下を追記して、コマンドを実行します。

```
STATIC_ROOT = os.path.join(BASE_DIR, 'static/')
```

```
$ docker-compose run web ./manage.py collectstatic
```

動作確認

コンテナを立ち上げて、ブラウザで `localhost:8000` にアクセスしてページが正しく表示されるか確認します。

```
$ docker-compose up
```

以下の画面が表示されれば成功です。

The install worked successfully! Congratulations!

You are seeing this page because DEBUG=True is in your settings file and you have not configured any URLs.

余談（環境構築後のプロジェクトの諸設定）

言語とタイムゾーンの設定

日本語化とタイムゾーンを東京に設定するには、 `src/mysite/settings.py` を以下のように編集します。

```
LANGUAGE_CODE = 'ja-JP'
TIME_ZONE = 'Asia/Tokyo'
```

アプリケーションの追加

以下のコマンドでアプリを作成し、 `src/mysite/settings.py` に作成したアプリを追記します。

```
$ docker-compose run web ./manage.py startapp <アプリ名>
```

```
INSTALLED_APPS = [
     'django.contrib.admin',
     'django.contrib.auth',
     'django.contrib.contenttypes',
     'django.contrib.sessions',
     'django.contrib.messages',
     'django.contrib.staticfiles',
     '<アプリ名>',
 ]
```

その他の参考サイト

nmmmk.hatenablog.com

docs.docker.com

blog.moo-channel.net