

Створення Django-додатка для Docker

Цей документ детально пояснює, як підготувати Django-проект до роботи з Docker, що є розширеною версією пункту 1 з нашого основного посібника.

Ми створимо не просто порожній проект, а мінімальний, але повністю функціональний додаток, що використовує базу даних PostgreSQL.

1. Структура проекту

Створіть наступну структуру каталогів на вашій локальній машині:

```
django_project/  
├── web/  
│   ├── manage.py  
│   ├── requirements.txt  
│   └── myapp/  
│       ├── __init__.py  
│       ├── settings.py  
│       ├── urls.py  
│       ├── views.py  
│       └── templates/  
│           └── myapp/  
│               └── index.html
```

2. Налаштування файлів та залежностей

Файл web/requirements.txt

Цей файл містить усі Python-бібліотеки, необхідні для нашого проекту. psycopg2-binary — це адаптер для підключення до бази даних PostgreSQL.

```
Django>=4.2,<5.0  
psycopg2-binary  
gunicorn
```

Файл web/Dockerfile

Це інструкція для створення образу нашого Django-додатка. Вона залишається такою, як

і в основному посібнику, оскільки є універсальною.

```
FROM python:3.9-slim

WORKDIR /usr/src/app

COPY requirements.txt ./
RUN pip install --no-cache-dir -r requirements.txt

COPY . .

CMD ["gunicorn", "myapp.wsgi:application", "--bind", "0.0.0.0:8000"]
```

3. Файли Django-додатка

Файл `web/myapp/settings.py`

Тут ми налаштовуємо Django-проект. Найважливіший аспект — це підключення до бази даних PostgreSQL. Ми використовуємо змінні середовища, щоб не зберігати чутливі дані в коді.

```
import os
from pathlib import Path

BASE_DIR = Path(__file__).resolve().parent.parent

SECRET_KEY = 'your-secret-key'

DEBUG = os.environ.get('DEBUG', 'False') == 'True'

ALLOWED_HOSTS = ['*']

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'myapp', # Додаємо наш додаток
```

```

]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'myapp.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR, 'templates')], # Вказуємо шлях до
темплейтів
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'myapp.wsgi.application'

# Налаштування бази даних PostgreSQL
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': os.environ.get('POSTGRES_DB', 'django_db'),
        'USER': os.environ.get('POSTGRES_USER', 'admin'),
        'PASSWORD': os.environ.get('POSTGRES_PASSWORD', 'password'),
        'HOST': 'postgres', # Важливо! Використовуємо ім'я сервісу з
docker-compose.yml

```

```

        'PORT': 5432,
    }
}

AUTH_PASSWORD_VALIDATORS = [
    {'NAME':
     'django.contrib.auth.password_validation.UserAttributeSimilarityValidator'},
    {'NAME':
     'django.contrib.auth.password_validation.MinimumLengthValidator'},
    {'NAME':
     'django.contrib.auth.password_validation.CommonPasswordValidator'},
    {'NAME':
     'django.contrib.auth.password_validation.NumericPasswordValidator'},
]

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_TZ = True

STATIC_URL = 'static/'

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'

```

Файл web/myapp/urls.py

Тут ми визначаємо URL-маршрути для нашого додатка.

```

from django.contrib import admin
from django.urls import path
from .views import index

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', index, name='index'), # Визначаємо маршрут для головної
    сторінки
]

```

```
]
```

Файл web/myapp/views.py

Цей файл містить нашу логіку відображення.

```
from django.shortcuts import render

def index(request):
    """
    Головна сторінка нашого додатку.
    """
    context = {
        'message': 'Ваш Django-додаток успішно запущено в Docker!',
    }
    return render(request, 'myapp/index.html', context)
```

Файл web/myapp/templates/myapp/index.html

Це наш HTML-шаблон, який буде відображатися на головній сторінці.

```
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <title>Django & Docker</title>
</head>
<body>
    <h1>Вітаємо!</h1>
    <p>{{ message }}</p>
    <p>Це простий Django-додаток, який працює в контейнері Docker.</p>
    <p>Підключення до бази даних PostgreSQL налаштовано і працює
коректно.</p>
</body>
</html>
```

Висновок

Ці файли становлять основу Django-проєкту, повністю готового до контейнеризації. Зверніть увагу, що в `settings.py` ми використовуємо `HOST: 'postgres'`, що дозволяє контейнеру Django підключатися до контейнера бази даних за його іменем сервісу, визначеним у `docker-compose.yml`.

Ця конфігурація дозволяє вам легко запускати додаток як локально, так і на віддаленому сервері, використовуючи той самий код та налаштування.