**Django + Docker + PostgreSQL + Redis + Elasticsearch**

1. Install Docker
2. Django Hello World

Cd folder create project

Example:

Mkdir code && cd code

Mkdir hello && cd hello

Pipenv install django==version

Pipenv shell

Django-admin startproject hello\_project .

Python manage.py migrate

Python manage.py runserver

1. Pages app

Python manage.py startapp pages

* Config settings.py

# hello\_project/settings.py

INSTALLED\_APPS = [

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

'pages.apps.PagesConfig', # new

]

* Urls.py

# hello\_project/urls.py

from django.contrib import admin

from django.urls import path, include # new

urlpatterns = [

path('admin/', admin.site.urls),

path('', include('pages.urls')), # new

]

* Views

# pages/views.py

from django.http import HttpResponse

def home\_page\_view(request):

return HttpResponse('Hello, World!')

* Create pages/urls.py

Touch pages/urls.py

# pages/urls.py

from django.urls import path

from .views import home\_page\_view

urlpatterns = [

path('', home\_page\_view, name='home')

]

* Python manage.py runserver
* Exit

(exit env)

1. Docker

* Touch Dockerfile

FROM python:3.7

# Set environment variables

ENV PYTHONDONTWRITEBYTECODE 1

ENV PYTHONUNBUFFERED 1

# Set work directory

WORKDIR /code

# Install dependencies

COPY Pipfile Pipfile.lock /code/

RUN pip install pipenv && pipenv install --system

# Copy project

COPY . /code/

* Docker build .
* Touch docker-compose.yml

docker-compose.yml

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

version: '3.7'

services:

web:

build: .

command: python /code/manage.py runserver 0.0.0.0:8000

volumes:

- .:/code

ports:

- 8000:8000

- docker-compose up

- docker-compose down

1. Git

Git init

Git status

Git add .

Git commit -m ‘ch1’

1. Postgresql

Install postgresql

====================================================================

1. Start project

Mkdir baoanhn && cd baoanhn

Pipenv install django==5.0.6 psycopg2-binary

Pipenv shell

Django-admin startproject baoanhn .

Python manage.py runserver

Exit

Touch Dockerfile

Touch docker-compose.yml

Dockerfile

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

# Pull base image

FROM python:3.11

# Set environment variables

ENV PYTHONDONTWRITEBYTECODE 1

ENV PYTHONUNBUFFERED 1

# Set work directory

WORKDIR /code

# Install dependencies

COPY Pipfile Pipfile.lock /code/

RUN pip install pipenv && pipenv install --system

# Copy project

COPY . /code/

docker-compose.yml

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

version: '3.11'

services:

web:

build: .

command: python /code/manage.py runserver 0.0.0.0:8000

volumes:

- .:/code

ports:

- 8000:8000

depends\_on:

- db

db:

image: postgres:16

volumes:

- postgres\_data:/var/lib/postgresql/data/

environment:

      POSTGRES\_DB: postgres

      POSTGRES\_USER: postgres

      POSTGRES\_PASSWORD: 123456

volumes:

postgres\_data:

* Docker-compose up -d –build
* Config settings

DATABASES = {

'default': {

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

'NAME': 'postgres',

'USER': 'postgres',

'PASSWORD': 'postgres',

'HOST': 'db',

'PORT': 5432

}

}

* Users:

Docker-compose exec web python manage.py startapp users

# users/models.py

from django.contrib.auth.models import AbstractUser

from django.db import models

class CustomUser(AbstractUser):

pass

* Configs settings

INSTALLED\_APPS = [

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

# Local

'users.apps.UsersConfig', # new

]

...

AUTH\_USER\_MODEL = 'users.CustomUser' # new

* docker-compose exec web python manage.py makemigrations users
* docker-compose exec web python manage.py migrate

1. Custom User Forms

Touch users/forms.py

* Config users/forms.py

# users/forms.py

from django.contrib.auth import get\_user\_model

from django.contrib.auth.forms import UserCreationForm, UserChangeForm

class CustomUserCreationForm(UserCreationForm):

class Meta:

model = get\_user\_model()

fields = ('email', 'username',)

class CustomUserChangeForm(UserChangeForm):

class Meta:

model = get\_user\_model()

fields = ('email', 'username',)

1. Custom user admin

# users/admin.py

from django.contrib import admin

from django.contrib.auth import get\_user\_model

from django.contrib.auth.admin import UserAdmin

from .forms import CustomUserCreationForm, CustomUserChangeForm

CustomUser = get\_user\_model()

class CustomUserAdmin(UserAdmin):

add\_form = CustomUserCreationForm

form = CustomUserChangeForm

model = CustomUser

list\_display = ['email', 'username',]

admin.site.register(CustomUser, CustomUserAdmin)

1. SuperUser

Docker-compose exec web python manage.py createsuperuser

1. Unit test

# users/tests.py

from django.contrib.auth import get\_user\_model

from django.test import TestCase

class CustomUserTests(TestCase):

def test\_create\_user(self):

User = get\_user\_model()

user = User.objects.create\_user(

username='will',

email='will@email.com',

password='testpass123'

)

self.assertEqual(user.username, 'will')

self.assertEqual(user.email, 'will@email.com')

self.assertTrue(user.is\_active)

self.assertFalse(user.is\_staff)

self.assertFalse(user.is\_superuser)

def test\_create\_superuser(self):

User = get\_user\_model()

admin\_user = User.objects.create\_superuser(

username='superadmin',

email='superadmin@email.com',

password='testpass123'

)

self.assertEqual(admin\_user.username, 'superadmin')

self.assertEqual(admin\_user.email, 'superadmin@email.com')

self.assertTrue(admin\_user.is\_active)

self.assertTrue(admin\_user.is\_staff)

self.assertTrue(admin\_user.is\_superuser)

* Docker-compose exec web python manage.py test

1. Git

Git init

Git status

Git add .

Git commit -m “baoanhn”

1. Pages App

* Docker-compose exec web python manage.py startapp pages

# bookstore\_project/settings.py

INSTALLED\_APPS = [

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

# Local

'users.apps.UsersConfig',

'pages.apps.PagesConfig', # new

]

TEMPLATES = [

{

...

'DIRS': [os.path.join(BASE\_DIR, 'templates')], # new

...

}

]

* Templates

Mkdir templates

Touch templates/\_base.html

Touch templates/home.html

<!-- templates/\_base.html -->

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>{% block title %}Bookstore{% endblock title %}</title>

</head>

<body>

<div class="container">

{% block content %}

{% endblock content %}

</div>

</body>

</html>

<!-- templates/home.html -->

{% extends '\_base.html' %}

{% block title %}Home{% endblock title %}

{% block content %}

<h1>Homepage</h1>

{% endblock content %}

# bookstore\_project/urls.py

from django.contrib import admin

from django.urls import path, include # new

urlpatterns = [

path('admin/', admin.site.urls),

path('', include('pages.urls')), # new

]

* Touch pages/urls.py

# pages/urls.py

from django.urls import path

from .views import HomePageView

urlpatterns = [

path('', HomePageView.as\_view(), name='home'),

]

# pages/views.py

from django.views.generic import TemplateView

class HomePageView(TemplateView):

template\_name = 'home.html'

* Test

# pages/tests.py

from django.test import SimpleTestCase

from django.urls import reverse

class HomepageTests(SimpleTestCase):

def test\_homepage\_status\_code(self):

response = self.client.get('/')

self.assertEqual(response.status\_code, 200)

def test\_homepage\_url\_name(self):

response = self.client.get(reverse('home'))

self.assertEqual(response.status\_code, 200)

* Docker-compose exec web python manage.py test
* Testing Templates

# pages/tests.py

from django.test import SimpleTestCase

from django.urls import reverse

class HomepageTests(SimpleTestCase):

def test\_homepage\_status\_code(self):

response = self.client.get('/')

self.assertEqual(response.status\_code, 200)

def test\_homepage\_url\_name(self):

response = self.client.get(reverse('home'))

self.assertEqual(response.status\_code, 200)

def test\_homepage\_template(self): # new

response = self.client.get('/')

self.assertTemplateUsed(response, 'home.html')

* Docker-compose exec web python manage.py test pages
* Test HTML

# pages/tests.py

from django.test import SimpleTestCase

from django.urls import reverse, resolve

from .views import HomePageView

class HomepageTests(SimpleTestCase):

def test\_homepage\_status\_code(self):

response = self.client.get('/')

self.assertEqual(response.status\_code, 200)

def test\_homepage\_url\_name(self):

response = self.client.get(reverse('home'))

self.assertEqual(response.status\_code, 200)

def test\_homepage\_template(self):

response = self.client.get('/')

self.assertTemplateUsed(response, 'home.html')

def test\_homepage\_contains\_correct\_html(self): # new

response = self.client.get('/')

self.assertContains(response, 'Homepage')

def test\_homepage\_does\_not\_contain\_incorrect\_html(self): # new

response = self.client.get('/')

self.assertNotContains(

response, ‘Hi there! I should not be on the page. ’)

* Docker-sompose exec web python manage.py test