Menu

Contents

[1. Cài đặt 2](#_Toc528074269)

[2. Django – Tạo project Django 2](#_Toc528074270)

[1. Tạo project 2](#_Toc528074271)

[Run server 2](#_Toc528074272)

[Tạo web app 2](#_Toc528074273)

[2. Tạo view 3](#_Toc528074274)

[3. Thiết lập cơ sở dữ liệu 5](#_Toc528074275)

[3. Tạo mô hình dữ liệu cho ứng dụng 5](#_Toc528074276)

[4. Tạo bảng từ mô hình dữ liệu 6](#_Toc528074277)

[Thao tác với các bảng 6](#_Toc528074278)

# Cài đặt

Lệnh cài đặt

pip install Django

Nếu cài theo phiên bản

pip install Django==1.9.4

Kiểm tra phiên bản

Code

import django

print(django.get\_version())

# Django – Tạo project Django

## Tạo project

Lệnh tạo

django-admin startproject mysite

Ở đây mysite là project web cần tạo. Cấu trúc thư mục lúc này sẽ như sau:

mysite/

    manage.py

    mysite/

        \_\_init\_\_.py

        settings.py

        urls.py

        wsgi.py

Mỗi ứng dụng web Django sẽ chạy riêng 1 server nên có thể đặt code web ở bất kì đâu (khác với PHP, JSP… cần cài webserver)

## Run server

Lệnh chạy

python manage.py runserver

manage.py là file cấu hình server cho mỗi ứng dụng web. Mặc định ứng dụng web python sẽ chạy trên cổng 8000.

Để đổi cổng chạy sang cổng khác (vd 8080) thực hiện như sau:

python manage.py runserver 8080

## Tạo web app

Phần trên là tạo webserver cho website, để tạo ứng dụng web, thực hiện:

python manage.py startapp polls

Với polls là ứng dụng cần tạo ra

Cấu trúc ứng dụng web sẽ như sau:

polls/

    \_\_init\_\_.py

    admin.py

    apps.py

    migrations/

        \_\_init\_\_.py

    models.py

    tests.py

    views.py

Project web khác với app web là project chứa nhiều app, mỗi app thực hiện một chức năng, công việc riêng biệt.

## Tạo view

polls/views.py

from django.http import HttpResponse

def index(request):

    response = HttpResponse()

    response.write("<h1>Welcome</h1>")

    response.write("This is the polls app")

    return response

Bên trong file views.py này chúng ta viết hàm index, hàm này trả về một đối tượng HttpResponse.

Để gọi được tới hàm index này và lấy nội dung HTML thì chúng ta phải tạo đường dẫn URL tới file này.

polls/

    \_\_init\_\_.py

    admin.py

    apps.py

    migrations/

        \_\_init\_\_.py

    models.py

    tests.py

    urls.py

    views.py

Để tạo URL chúng ta tạo một file với tên urls.py bên trong thư mục polls.

polls/urls.py

from django.conf.urls import url

from . import views

 urlpatterns = [

     url(r'^$', views.index, name='index'),

]

Django sẽ tìm thông tin về các url trong list urlpatterns, list này chứa các đối tượng url, mỗi url bao gồm 3 tham số :

- tham số thứ nhất là biểu thức chính quy (Regex – Regular Expression),

- tham số thứ hai là tên module và tên hàm sẽ trả về nội dung HTML, ở đây là hàm index trong module views (file views.py),

- tham số thứ 3 là tên biến toàn cục trong toàn bộ ứng dụng, trong đó 2 tham số đầu tiên là bắt buộc phải có.

Tiếp theo chúng ta phải tạo đường dẫn đến ứng dụng polls này, chúng ta sẽ tạo đường dẫn cho từng ứng dụng trong file mysite/urls.py.

mysite/urls.py

from django.conf.urls import include, url

from django.contrib import admin

urlpatterns = [

    url(r'^polls/', include('polls.urls')),

    url(r'^admin/', admin.site.urls),

]

Bên trong file mysite/urls.py cũng có một list urlpatterns, list này chứa các đối tượng url dẫn đến các ứng dụng web khác mà chúng ta viết.

Ở đây tham số thứ 2 chúng ta không chỉ đến một hàm index nào đó trong một module nào đó, mà chúng ta chỉ đến một module có chứa đối tượng urlpatterns khác, trong trường  hợp này là polls.urls, nên chúng ta phải dùng đến hàm include(), đối với các url chỉ đến một module url khác thì chuỗi regex không kết thúc bằng dấu $, hay nói cách khác là không có kết thúc.

Khi chúng ta gõ localhost:8000 thì django sẽ đọc các url trong file mysite/urls.py, trong file này chúng ta có một đường dẫn đến admin/, đường dẫn này trỏ đến hàm get\_url() trong module admin.site của django, do đó bạn có thể gõ localhost:8000/admin và django sẽ trả về một trang đăng nhập mẫu (chúng ta sẽ tìm hiểu về trang admin này sau), url thứ hai là đường dẫn polls/ trỏ đến một file urls.py khác ở module polls.urls do chúng ta tự viết, file này lại chứa một list urlpatterns khác chứa các đường dẫn đến các hàm trả về HTML riêng của chúng, thế nên chúng ta có thể gõ localhost:8000/polls, và django sẽ trả về trang HTML tạo ra từ hàm index() trong module polls.urls.

# Thiết lập cơ sở dữ liệu

Mặc định là SQLite. Nếu thay đổi vào file settings.py trong myweb để thay đổi.

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.sqlite3',

'NAME': os.path.join(BASE\_DIR, 'db.sqlite3'),

}

}

* ENGINE: tên module dành cho từng CSDL, mặc định sử dụng SQLite
  + django.db.backends.sqlite3 – cơ sở dữ liệu  SQLite
  + django.db.backends.postgresql – cơ sở dữ liệu PostgreSQL
  + django.db.backends.mysql – cơ sở dữ liệu MySQL
  + django.db.backends.oracle – cơ sở dữ liệu Oracle
* NAME: tên CSDL, mặc định là file db.sqlite3 được tạo ra ở thư mục gốc của server.

List INSTALLED\_APPS là các ứng dụng thường được sử dụng khi chạy web. Lệnh:

python manage.py migrate

sẽ tìm các module trong list này và tạo các table tương ứng.

## Tạo mô hình dữ liệu cho ứng dụng

polls/models.py

from django.db import models

class Question(models.Model):

    question\_text = models.CharField(max\_length=200)

    pub\_date = models.DateTimeField('date published')

class Choice(models.Model):

    question = models.ForeignKey(Question, on\_delete=models.CASCADE)

    choice\_text = models.CharField(max\_length=200)

    votes = models.IntegerField(default=0)

Mỗi mô hình được tạo ra phải thừa kế từ lớp: django.db.models.Model

## Tạo bảng từ mô hình dữ liệu

Sau khi đã tạo các lớp mô hình trong Python, Django sẽ nhìn các thuộc tính trong từng **lớp để tạo các bảng** tương ứng trong CSDL và **tạo thêm các lớp mới** trong Python cung cấp các hàm để thao tác với các bảng trong CSDL.

|  |
| --- |
| mysite/settings.py |
|  |

INSTALLED\_APPS = [

'polls.apps.PollsConfig',

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

]

Để Django tạo các bảng mới trong CSDL thì chúng ta phải khai báo trước trong list INSTALLED\_APPS đã bằng cách thêm dòng polls.apps.PollsConfig.

|  |
| --- |
|  |

Tiếp theo chạy lệnh makemigration polls để báo cho Django biết là bạn đã thay đổi một số mô hình, ở đây là thêm 2 lớp mới trong gói polls.

python manage.py makemigrations polls

Lệnh migrate để Django cập nhật lại CSDL. Lúc này CSDL chúng ta sẽ có thêm 2 bảng mới là polls\_choice và polls\_question (tên bảng được đặt theo cú pháp <tên package>\_<tên lớp>).

python manage.py migrate

## Thao tác với các bảng

Sau khi Django tạo các bảng, ta có thể thao tác các công việc thường dùng như thêm-xóa-sửa. Sử dụng shell của python thao tác với csdl, chạy lệnh:

python manage.py shell

Phương thức Question.objects.all() liệt kê toàn bộ đối tượng Question đang có trong CSDL

>>> from polls.models import Question, Choice

>>> Question.objects.all()

[]

Tạo một đối tượng Question và thiết lập pub\_date là ngày giờ hiện tại của máy tính

>>>from django.utils import timezone

>>>q = Question(question\_text="What's new?", pub\_date=timezone.now())

>>>q.save()

Khi viết lớp Question và Choice chúng ta không cung cấp thuộc tính id hay bất cứ thuộc tính nào để làm khóa chính, vì vậy khi tạo CSDL Django sẽ tự động thêm 1 thuộc tính id để làm khóa chính và thuộc tính này sẽ tự động tăng.

>>> q.id

1

Ta có thể thay đổi trực tiếp giá trị trên các đối tượng này sau đó gọi phương thức save() là dữ liệu trên CSDL sẽ được cập nhật

>>> q.question\_text

What's new?

>>> q.question\_text = "What's up?"

>>> q.save()

Nên override phương thức \_\_str\_\_() để khi in đối tượng ra người dùng có thể đọc được. Nếu không override phương thức này thì Question.objects.all() sẽ cho ra kết quả không đọc được.

polls/models.py

from django.db import models

class Question(models.Model):

    question\_text = models.CharField(max\_length=200)

    pub\_date = models.DateTimeField('date published')

    def \_\_str\_\_(self):

        return self.question\_text

class Choice(models.Model):

    question = models.ForeignKey(Question, on\_delete=models.CASCADE)

    choice\_text = models.CharField(max\_length=200)

    def \_\_str\_\_(self):

        return self.choice\_text

Chạy kết quả khi có hàm \_\_str\_\_. Lúc này phương thức objects.all() sẽ gọi đến phương thức \_\_str\_\_().

>>> from polls.models import Question, Choice

>>> Question.objects.all()

[<Question: What's up?>]

Lọc các bản ghi theo thuộc tính bằng phương thức filter().

>>> Question.objects.filter(id=1)

[<Question: What's up?>]

>>> Question.objects.filter(question\_text\_\_startswith='What')

[<Question: What's up?>]

Ta cũng có thể lấy từng bản ghi đơn lẻ bằng phương thức get(), ở trên chúng ta lấy theo khóa chính pk (Primary Key).

>>> q = Question.objects.get(pk=1)

Bất cứ bảng nào có một bảng khác chứa khóa ngoại tham chiếu đến khóa chính của nó đều sẽ có một thuộc tính được tạo tự động là một tập hợp các đối tượng của bảng kia. Ở đây bảng Choice chứa khóa ngoại tham chiếu đến bảng Question, do đó bảng Question sẽ có một danh sách các đối tượng Choice, chúng ta có thể lấy danh sách này qua choice\_set.all() (tên danh sách được đặt theo <tên bảng>\_set). Ở đây chúng ta vẫn chưa tạo đối tượng Choice nào nên danh sách trả về rỗng.

>>> q.choice\_set.all()

[]

Chúng ta tạo 3 đối tượng Choice bằng phương thức choice\_set.create().

>>> q.choice\_set.create(choice\_text='Not much', votes=0)

<Choice: Not much>

>>> q.choice\_set.create(choice\_text='The sky', votes=0)

<Choice: The sky>

>>> c = q.choice\_set.create(choice\_text='Just hacking again', votes=0)

Chúng ta có thể lấy số lượng các bản ghi trong bảng bằng phương thức count().

>>> q.choice\_set.all()

[<Choice: Not much>, <Choice: The sky>, <Choice: Just hacking again>]

>>> q.choice\_set.count()

3

Để xóa một bản ghi trong bảng thì chúng ta dùng phương thức delete().

>>> c = q.choice\_set.filter(choice\_text\_\_startswith='Just hacking')

>>> c.delete()