

Introduction to Django

吳柏翰 (Jerry Wu)

Top 5 Python Web Framework

- **Django**

- 目前Python主流框架，適合新創團隊使用，論壇資料多，適合快速打造應用。
- 框架完整是優點也是缺點
- 知名案例：[Instagram](#)、[edx](#)、[Pinterest](#)

- **Flask**

- 微框架，透過擴展來提升框架完整度，彈性高是優點也是缺點
- 適合有開發經驗的團隊使用。案例：[豆瓣小組](#)

- **Tornado**

- 異步非阻塞框架，適合用在需要響應非常快速的網站應用。
 - 同步：一件一件做，異步：每一件事情輪流做
 - 阻塞：一直等回覆，非阻塞：一直問

- **Bottle**

- 微框架，適合開發小型網站，不適合商業應用。

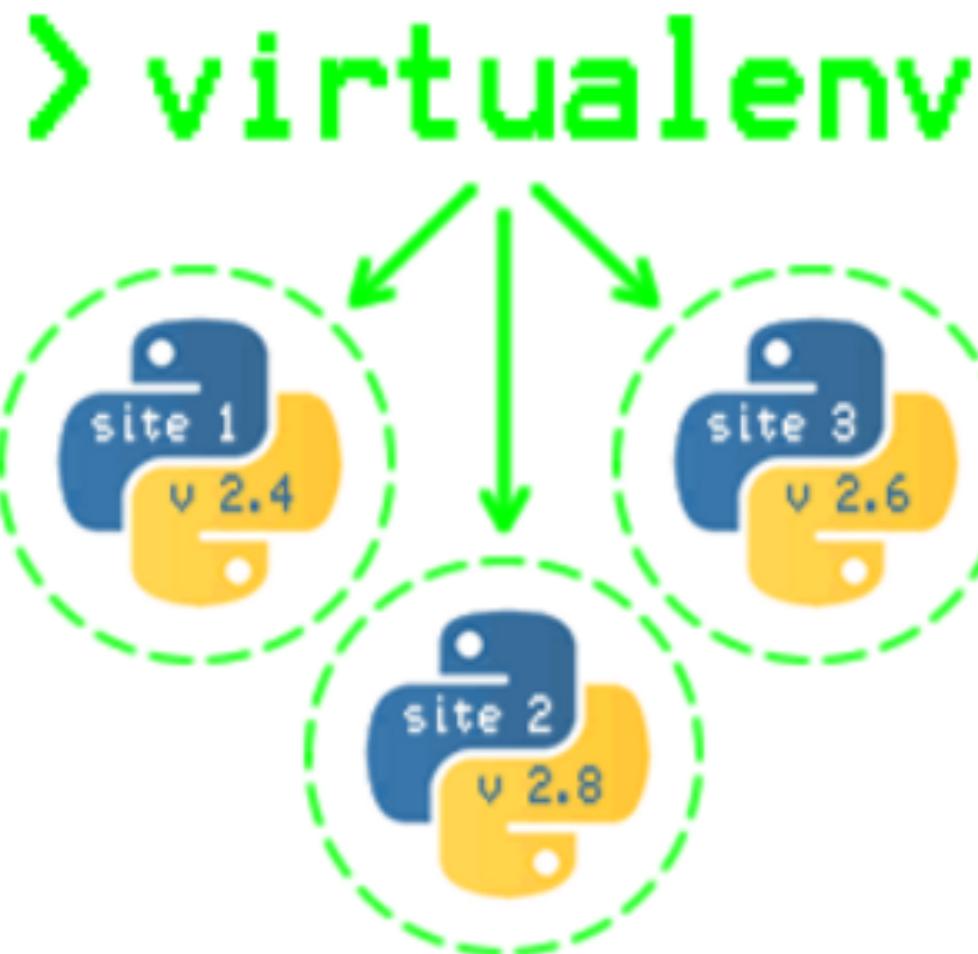
- **Web2py**

- 微框架，是由google在web.py二次開發，擴展性不佳。

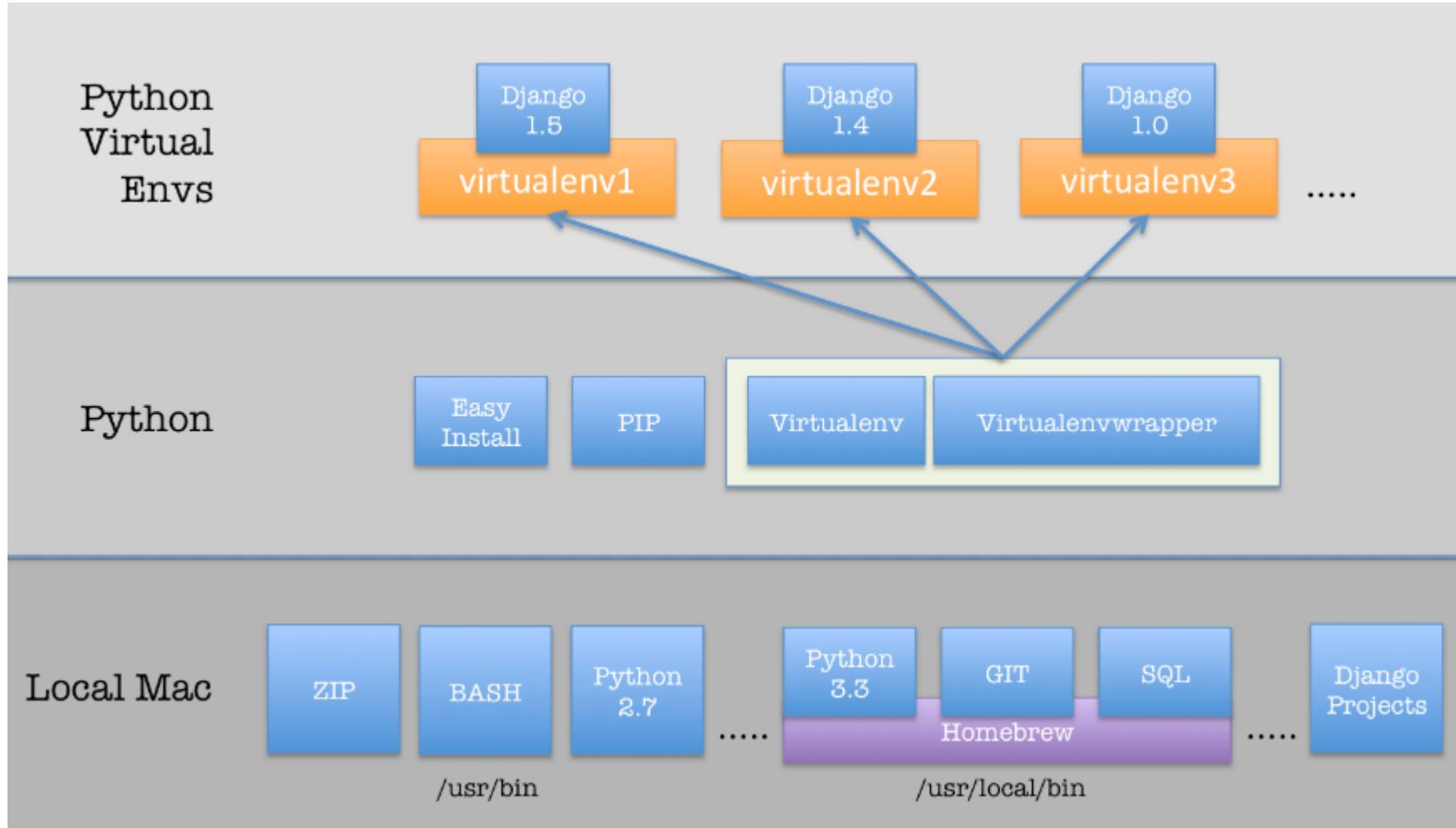
Joomla VS WP = Django VS Flask



Vitrualenv



Vitrualenv



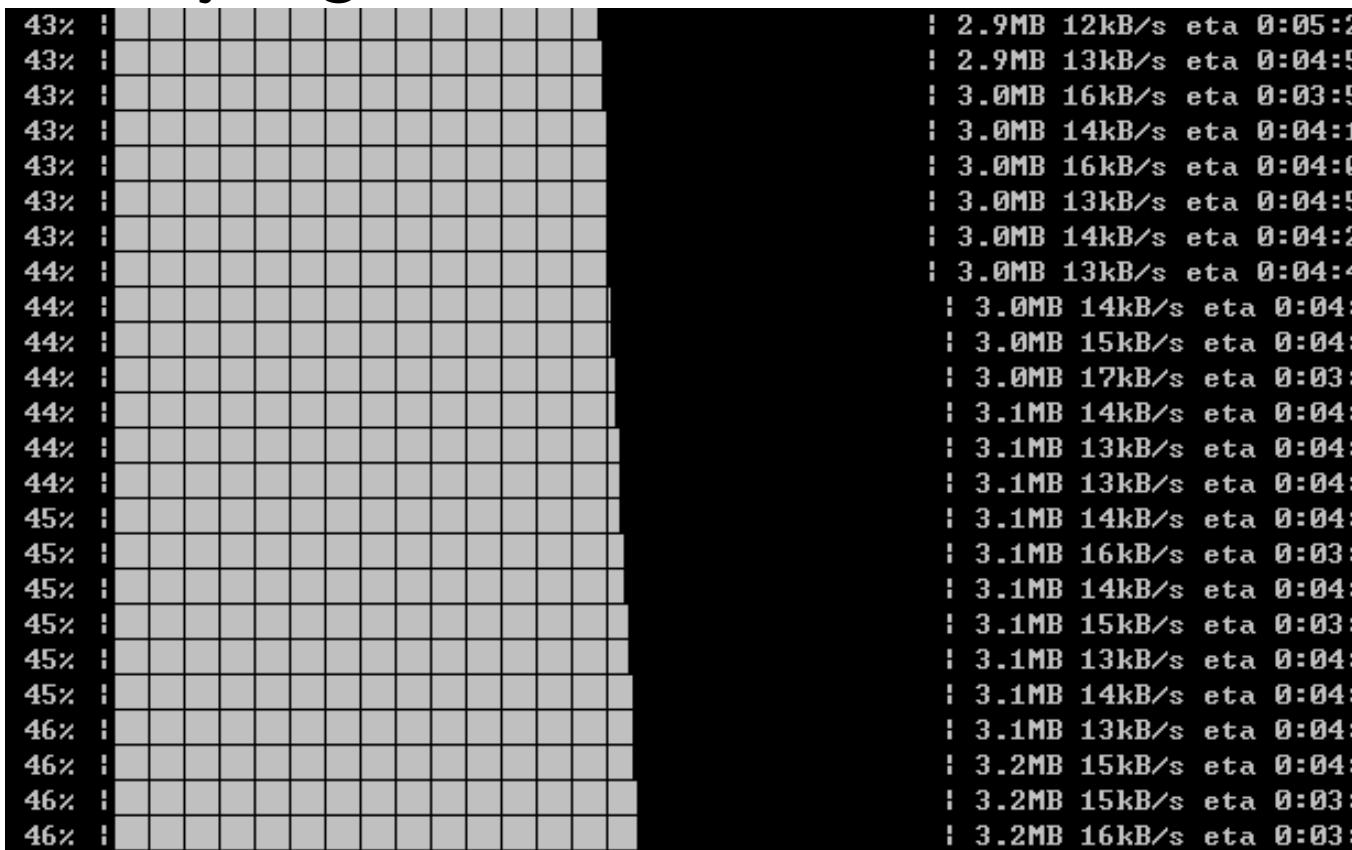
Vitrualenv

- pip install virtualenv
- pip3 install virtualenv
- easy_install virtualenv
- Mkdir class && cd class
- virtualenv . –p python3
- Virtualenv class
- WIN
 - Scripts\activate.bat
- MAC
 - source ./bin/activate
 - Deactivate

```
(m_django) c:\GitHub\mandarinx_django\m_django>
```

Install Django

```
pip3 install django==1.11
```



Install Django - Verifying

```
python -m django --version
```

```
(jerryweb) jerryde-MacBook-Pro:jerryweb jerry2017$ python -m django --version  
1.11
```

Part 1: Requests and responses

Creating a project

- django-admin startproject mysite
- python manage.py runserver

```
mysite/
    manage.py
mysite/
    __init__.py
    settings.py
    urls.py
    wsgi.py
```

Creating a project

- `manage.py`: A command-line utility that lets you interact with this Django project in various ways.
- `mysite/__init__.py`: An empty file that tells Python that this directory should be considered a Python package.
- `mysite/settings.py`: Settings/configuration for this Django project.
- `mysite/urls.py`: The URL declarations for this Django project.
- `mysite/wsgi.py`: An entry-point for WSGI-compatible web servers to serve your project.

Creating app

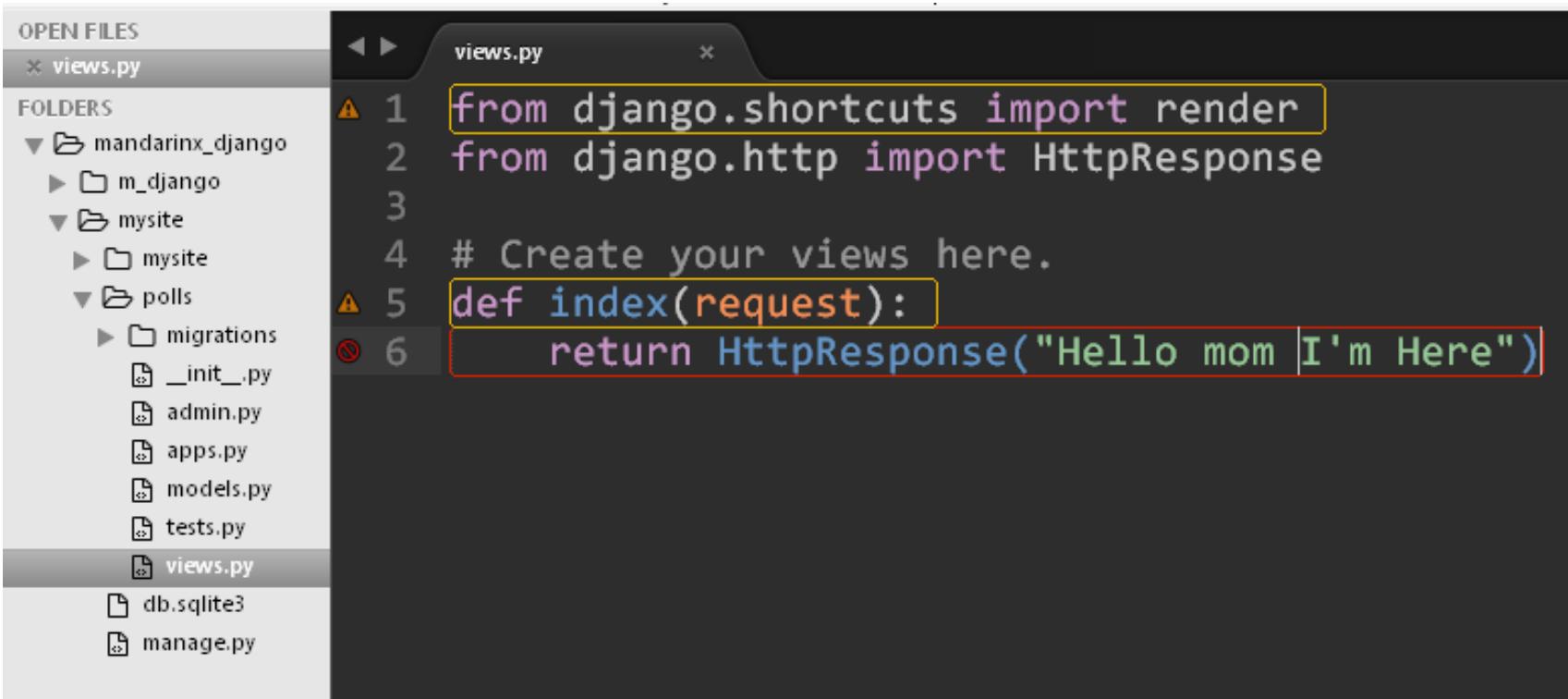
- What's the difference between a project and an app ?
- An app is a Web application that does something – e.g., a Weblog system, a database of public records or a simple poll app.
- A project is a collection of configuration and apps for a particular website. A project can contain multiple apps.
An app can be in multiple projects.

Creating app

- python manage.py startapp jerry

```
polls/
    __init__.py
    admin.py
    apps.py
    migrations/
        __init__.py
    models.py
    tests.py
    views.py
```

Write your first view



The screenshot shows a code editor interface with a dark theme. On the left is a sidebar titled "OPEN FILES" containing a list of files and folders:

- FOLDERS
 - mandarinx_django
 - m_django
 - mysite
 - mysite
 - polls
 - migrations
 - _init_.py
 - admin.py
 - apps.py
 - models.py
 - tests.py
- views.py
- db.sqlite3
- manage.py

The main editor window is titled "views.py" and contains the following Python code:

```
from django.shortcuts import render
from django.http import HttpResponseRedirect
# Create your views here.
def index(request):
    return HttpResponseRedirect("Hello mom I'm Here")
```

Write your first view

The screenshot shows a code editor interface with the following details:

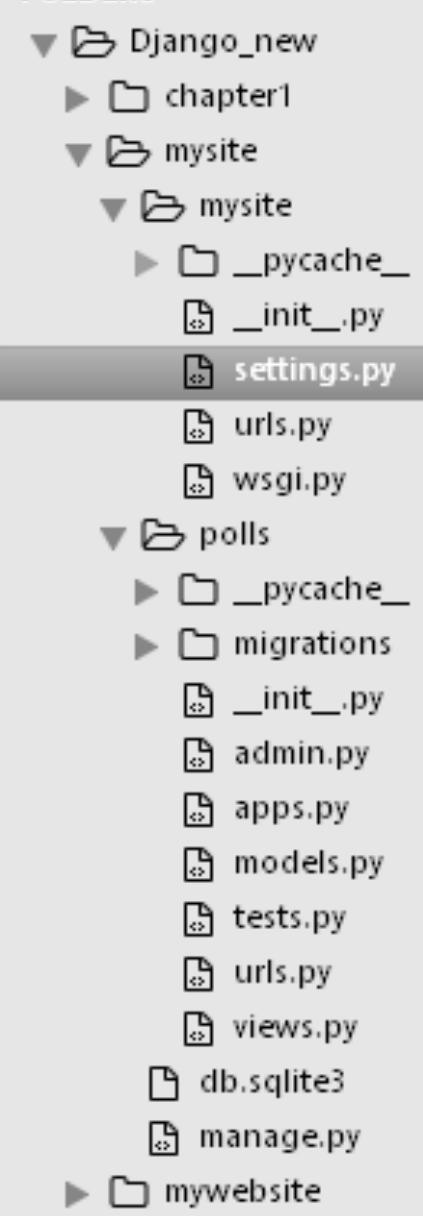
- OPEN FILES:** views.py, urls.py
- FOLDERS:** mandarinx_django, m_django, mysite, polls
- File Content (urls.py):**

```
1 from django.conf.urls import url
2
3 from . import views
4
5 urlpatterns = [
6     url(r'^$', views.index, name='index')
7 ]
```

The URL pattern at line 6 is highlighted with a red border.

Write your first view

```
3 The `urlpatterns` list routes URLs to views. For more information please see:
4     https://docs.djangoproject.com/en/dev/topics/http/urls/
5 Examples:
6 Function views
7     1. Add an import: from my_app import views
8         2. Add a URL to urlpatterns: url(r'^$', views.home, name='home')
9 Class-based views
10    1. Add an import: from other_app.views import Home
11        2. Add a URL to urlpatterns: url(r'^$', Home.as_view(), name='home')
12 Including another URLconf
13    1. Import the include() function: from django.conf.urls import url, include
14        2. Add a URL to urlpatterns: url(r'^blog/', include('blog.urls'))
15 """
16 from django.conf.urls import include, url
17 from django.contrib import admin
18
19 urlpatterns = [
▲20     url(r'^polls/', include('polls.urls')),
▲21     url(r'^admin/', admin.site.urls),
22 ]
23
```



```
INSTALLED_APPS = [
    'polls.apps.PollsConfig',
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]

# Internationalization
# https://docs.djangoproject.com/en/dev/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'Asia/Taipei'

USE_I18N = True

USE_L10N = True

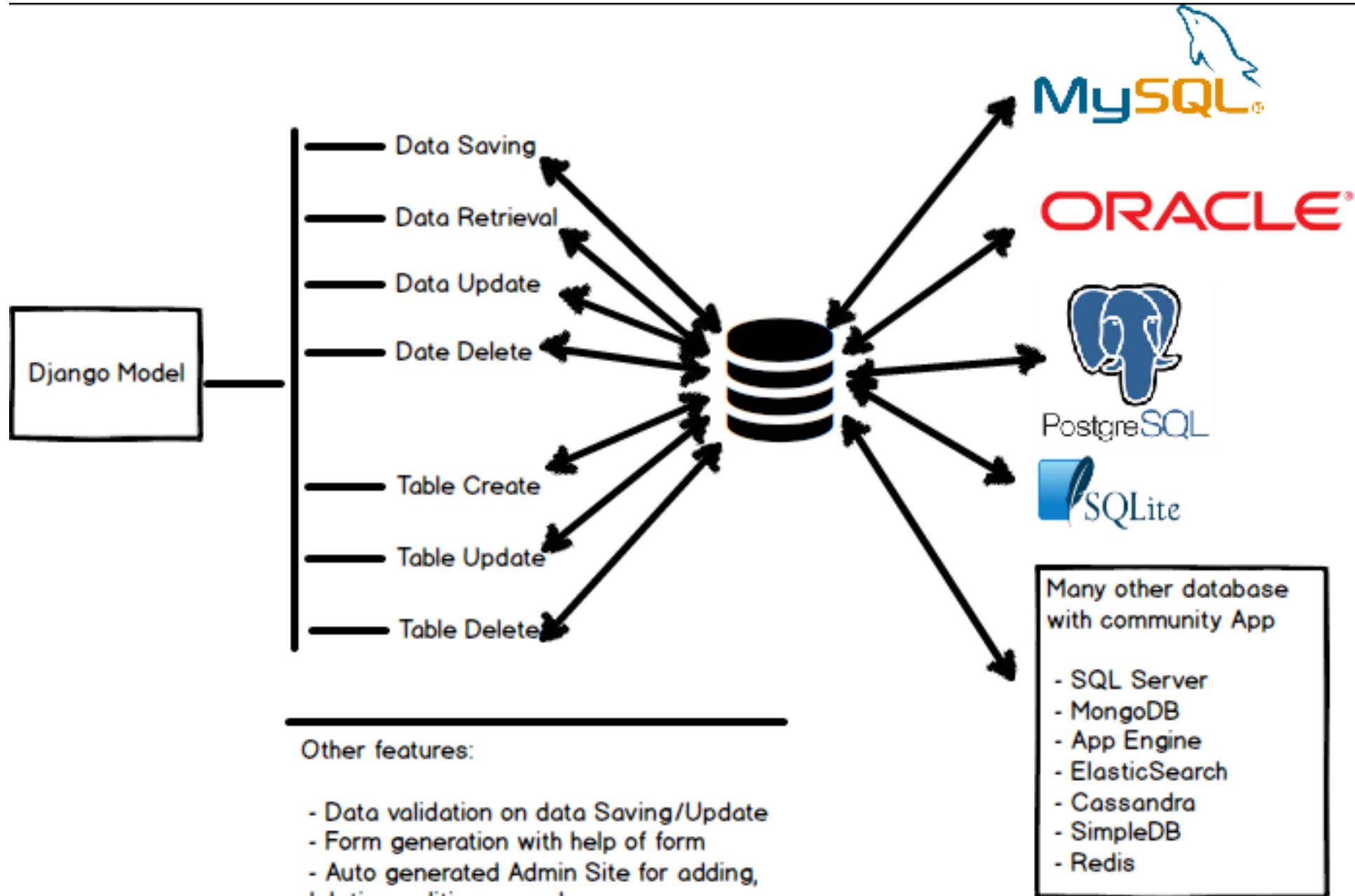
USE_TZ = True
```

Part 2: Models and the admin site

Part 2: Models and the admin site

```
$ python manage.py migrate
```

- The [migrate](#) command looks at the [INSTALLED_APPS](#) setting and creates any necessary database tables according to the database settings in your mysite/settings.py file and the database migrations shipped with the app.



File Edit Selection Find View Goto Tools Project Preferences Help Corona Editor

OPEN FILES

- views.py
- urls.py — polls
- urls.py — mysite
- settings.py
- models.py

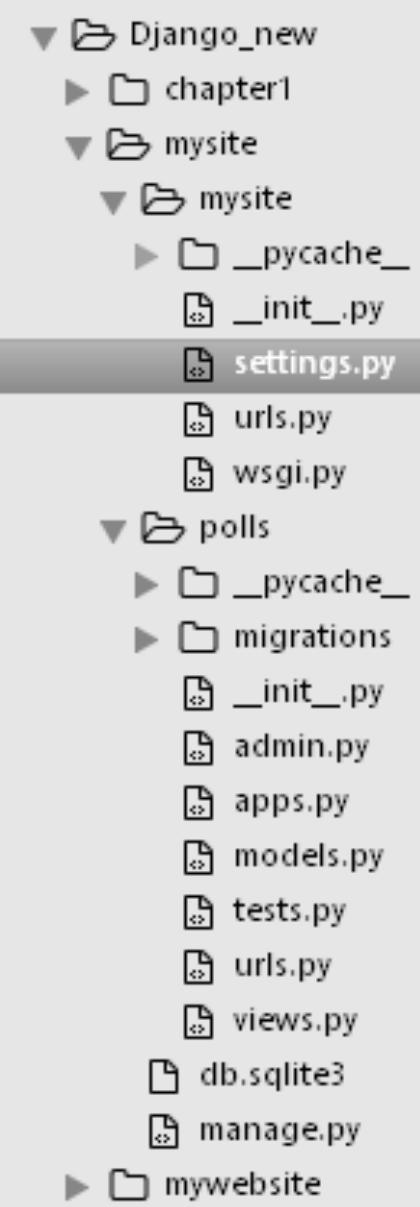
FOLDERS

- ▼ Django_new
 - ▶ chapter1
 - ▼ mysite
 - ▶ mysite
 - ▶ __pycache__
 - __init__.py
 - settings.py
 - urls.py
 - wsgi.py
 - ▼ polls
 - ▶ __pycache__
 - ▶ migrations
 - __init__.py
 - admin.py
 - apps.py
 - models.py
 - tests.py
 - urls.py
 - views.py
- ▶ mywebsite

```
1 from django.db import models
2
3 # Create your models here.
4 class Question(models.Model):
5     question_text = models.CharField(max_length=200)
6     pub_date = models.DateTimeField('date published')
7
8
9 class Choice(models.Model):
10    question = models.ForeignKey(Question, on_delete=models.CASCADE)
11    choice_text = models.CharField(max_length=200)
12    votes = models.IntegerField(default=0)
```



當一個模型對象的ForeignKey的關聯的對象被刪除時，默認情況下此對象也會一起被級聯刪除的。



```
INSTALLED_APPS = [
    'polls.apps.PollsConfig',
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]

# Internationalization
# https://docs.djangoproject.com/en/dev/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'Asia/Taipei'

USE_I18N = True

USE_L10N = True

USE_TZ = True
```

- `python manage.py makemigrations polls`
- `python manage.py sqlmigrate polls 0001`
- `python manage.py migrate`

```
-- Create model Choice
--

CREATE TABLE "polls_choice" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "choice_text" varchar(200) NOT NULL, "votes" integer NOT NULL);

-- Create model Question
--

CREATE TABLE "polls_question" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "question_text" varchar(200) NOT NULL, "pub_date" datetime NOT NULL);

-- Add field question to choice
--

ALTER TABLE "polls_choice" RENAME TO "polls_choice_old";
CREATE TABLE "polls_choice" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "choice_text" varchar(200) NOT NULL, "votes" integer NOT NULL, "question_id" integer NOT NULL REFERENCES "polls_question" ("id"));
INSERT INTO "polls_choice" ("question_id", "id", "choice_text", "votes") SELECT NULL, "id", "choice_text", "votes" FROM "polls_choice_old";
DROP TABLE "polls_choice_old";
CREATE INDEX "polls_choice_question_id_c5b4b260" ON "polls_choice" ("question_id");
COMMIT;
```

Playing with the API

- `python manage.py shell`
- `from polls.models import Question, Choice`
- `Question.objects.all()`
- `from django.utils import timezone`
- `q = Question(question_text="What's new", pub_date=timezone.now())`
- `q.save()`
- `q.id`
- `q.question_text`
- `q.pub_date`

Playing with the API

- q.question_text = "What's up?"
- q.save()
- Question.objects.all()

```
>>> Question.objects.all()
<QuerySet [<Question: Question object>]>
>>>
```

Playing with the API

```
from django.db import models

# Create your models here.
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)
    votes = models.IntegerField(default=0)

    def __str__(self):
        return self.choice_text
```

Playing with the API

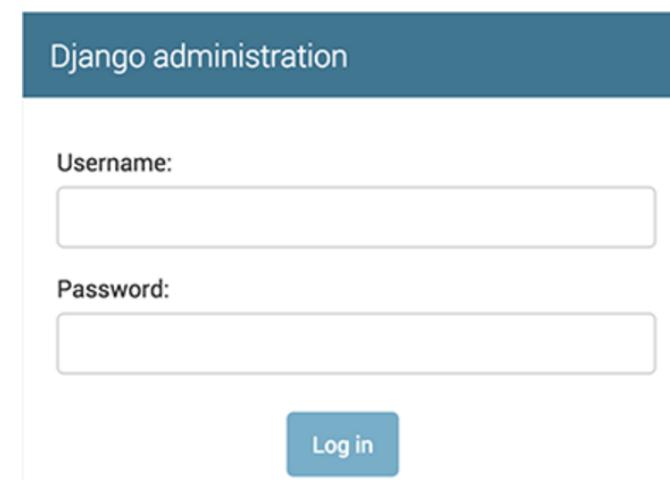
- python manage.py makemigrations polls
 - python manage.py migrate
 - python manage.py shell
-
- **from** polls.models **import** Question, Choice
 - Question.objects.all()

```
>>> from polls.models import Question, Choice  
>>> Question.objects.all()  
<QuerySet [<Question: What's up>]>
```

Creating an admin user

- python manage.py createsuperuser
- Username: admin
- Email address: admin@example.com
- Password: &1234567
- python manage.py runserver

Now, open a Web browser and go to "/admin/" on your local domain – e.g.,
<http://127.0.0.1:8000/admin/>. You should see the admin's login screen:



Django administration

WELCOME, **ADMIN**. [VIEW SITE / CHANGE PASSWORD / LOG OUT](#)

Site administration

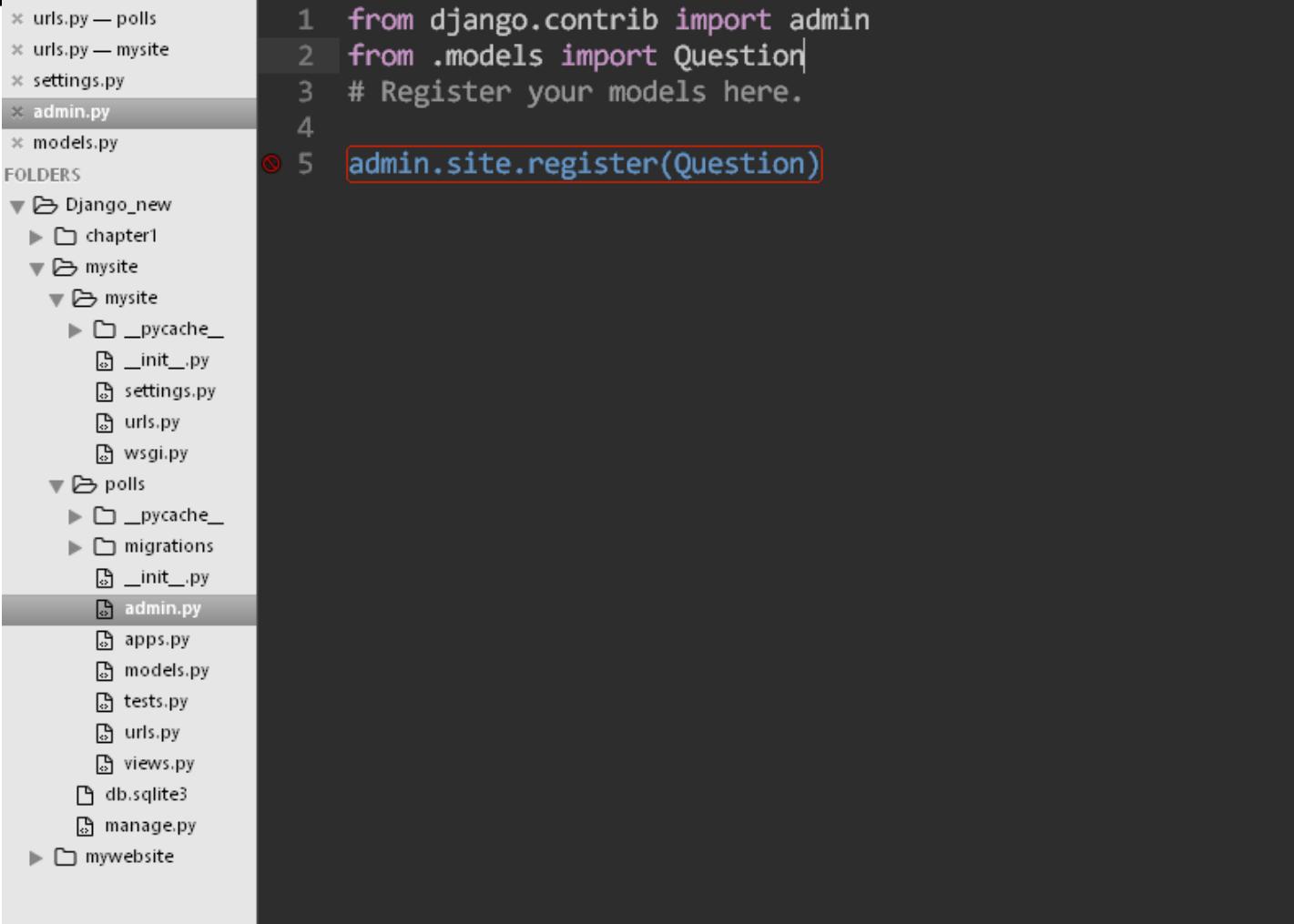
AUTHENTICATION AND AUTHORIZATION		
Groups	 Add	 Change
Users	 Add	 Change

Recent actions

My actions

None available

Make the poll app modifiable in the admin



The image shows a code editor interface with a sidebar displaying the project's directory structure. The structure includes a top-level folder 'mysite' containing 'Django_new', 'settings.py', 'urls.py', and 'wsgi.py'. Inside 'Django_new' are 'chapter1' and 'mysite'. The 'mysite' folder contains 'mysite' (which has '_pycache_'), 'polls' (which has '_pycache_'), and 'admin.py'. The 'polls' folder also contains 'migrations', 'apps.py', 'models.py', 'tests.py', 'urls.py', 'views.py', 'db.sqlite3', and 'manage.py'. A 'mywebsite' folder is also present at the bottom level.

```
1 from django.contrib import admin
2 from .models import Question
3 # Register your models here.
4
5 admin.site.register(Question)
```

Django administration

WELCOME, **ADMIN**. [VIEW SITE](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)

Site administration

AUTHENTICATION AND AUTHORIZATION

Groups

[+ Add](#) [Change](#)

Users

[+ Add](#) [Change](#)

POLLS

Questions

[+ Add](#) [Change](#)

Recent actions

My actions

None available

Z

Add question

Question text:

What's up?

Date published:

Date: 2017-03-25 [Today](#) | Time: 13:51:15 [Now](#) | [Save and add another](#)[Save and continue editing](#)**SAVE**

Change history: What's up?

DATE/TIME	USER	ACTION
March 25, 2017, 1:52 p.m.	admin	Added.
March 25, 2017, 1:53 p.m.	admin	No fields changed.
March 25, 2017, 1:53 p.m.	admin	No fields changed.

Part 3: Views and templates

Functions

- Question “index” page – displays the latest few questions.
- Question “detail” page – displays a question text, with no results but with a form to vote.
- Question “results” page – displays results for a particular question.
- Vote action – handles voting for a particular choice in a particular question.

polls/urls.py

```
from django.conf.urls import url
from . import views

urlpatterns = [
    # ex: /polls/
    url(r'^$', views.index, name='index'),
    # ex: /polls/5/
    url(r'^(?P<question_id>[0-9]+)/$', views.detail, name='detail'),
    # ex: /polls/5/results/
    url(r'^(?P<question_id>[0-9]+)/results/$', views.results, name='results'),
    # ex: /polls/5/vote/
    url(r'^(?P<question_id>[0-9]+)/vote/$', views.vote, name='vote'),
```

⑭]

^ 表示開始
\$ 表示結束
+ 表示前面的元素至少重複一次
() 用表示一部分模式
?P 代表匹配某個模式

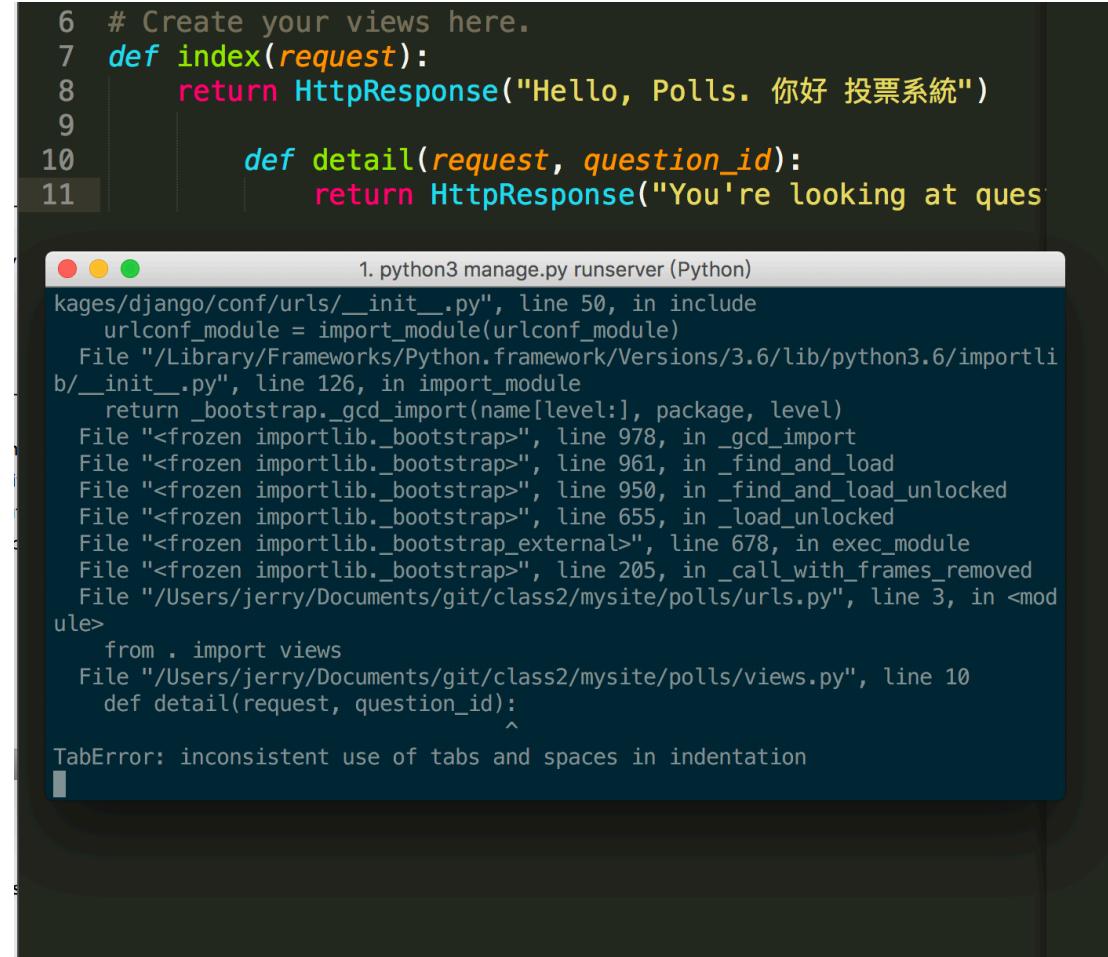
```
from django.db import models
# Create your models here.
class Question(models.Model):
    question_text = models.CharField(max_length=200)
    pub_date = models.DateTimeField('date published')
```

polls/views.py

The screenshot shows a code editor with a sidebar titled "OPEN FILES" and a main workspace with two tabs: "views.py" and "urls.py". The "views.py" tab is active, displaying Python code for a Django application. The code defines four view functions: index, detail, results, and vote. The "index" function returns a "Hello!" response. The "detail" function takes a question ID and returns a response about the question. The "results" function takes a question ID and returns the results of the question. The "vote" function takes a question ID and returns a response about voting. The code is highlighted with orange boxes around the function definitions.

```
from django.http import HttpResponseRedirect
# Create your views here.
def index(request):
    return HttpResponseRedirect("Hello!")
def detail(request, question_id):
    return HttpResponseRedirect("You're looking at question %s." % question_id)
def results(request, question_id):
    response = "You're looking at the results of question %s."
    return HttpResponseRedirect(response % question_id)
def vote(request, question_id):
    return HttpResponseRedirect("You're voting on question %s." % question_id)
```

常見錯誤

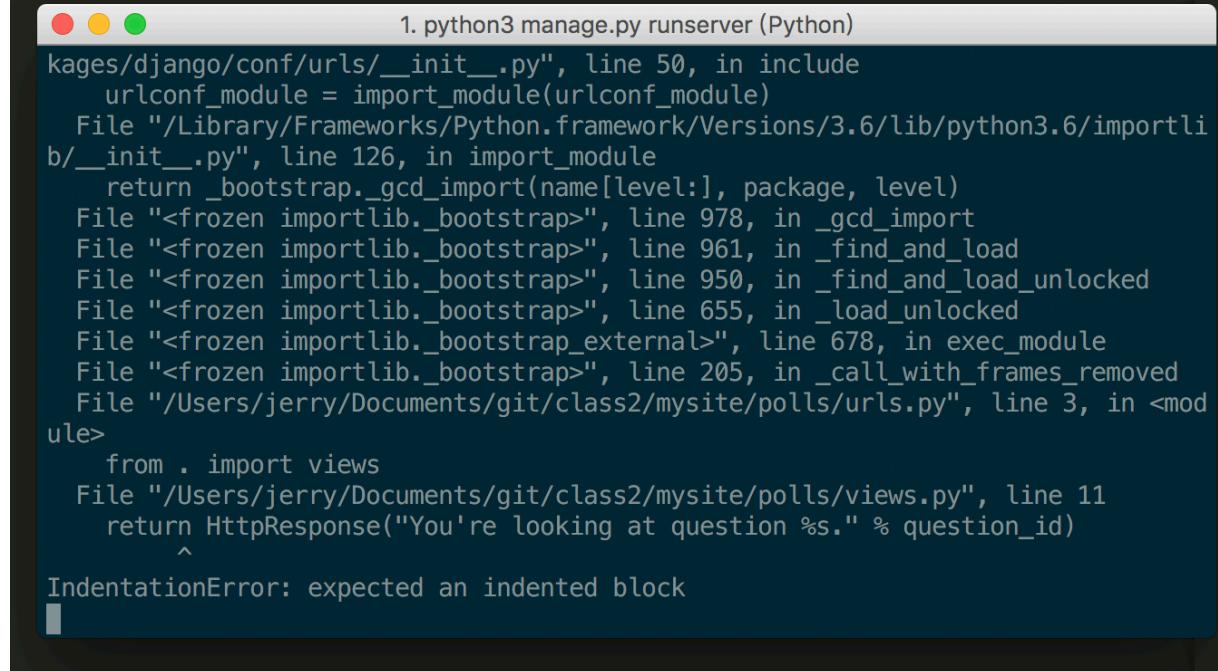


```
6 # Create your views here.
7 def index(request):
8     return HttpResponse("Hello, Polls. 你好 投票系統")
9
10    def detail(request, question_id):
11        return HttpResponse("You're looking at ques

1. python3 manage.py runserver (Python)
kages/django/conf/urls/__init__.py", line 50, in include
    urlconf_module = import_module(urlconf_module)
  File "/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/importlib/__init__.py", line 126, in import_module
      return _bootstrap._gcd_import(name[level:], package, level)
  File "<frozen importlib._bootstrap>", line 978, in _gcd_import
  File "<frozen importlib._bootstrap>", line 961, in _find_and_load
  File "<frozen importlib._bootstrap>", line 950, in _find_and_load_unlocked
  File "<frozen importlib._bootstrap>", line 655, in _load_unlocked
  File "<frozen importlib._bootstrap_external>", line 678, in exec_module
  File "<frozen importlib._bootstrap>", line 205, in _call_with_frames_removed
  File "/Users/jerry/Documents/git/class2/mysite/polls/urls.py", line 3, in <module>
      from . import views
  File "/Users/jerry/Documents/git/class2/mysite/polls/views.py", line 10
      def detail(request, question_id):
                                         ^
TabError: inconsistent use of tabs and spaces in indentation
```

常見錯誤

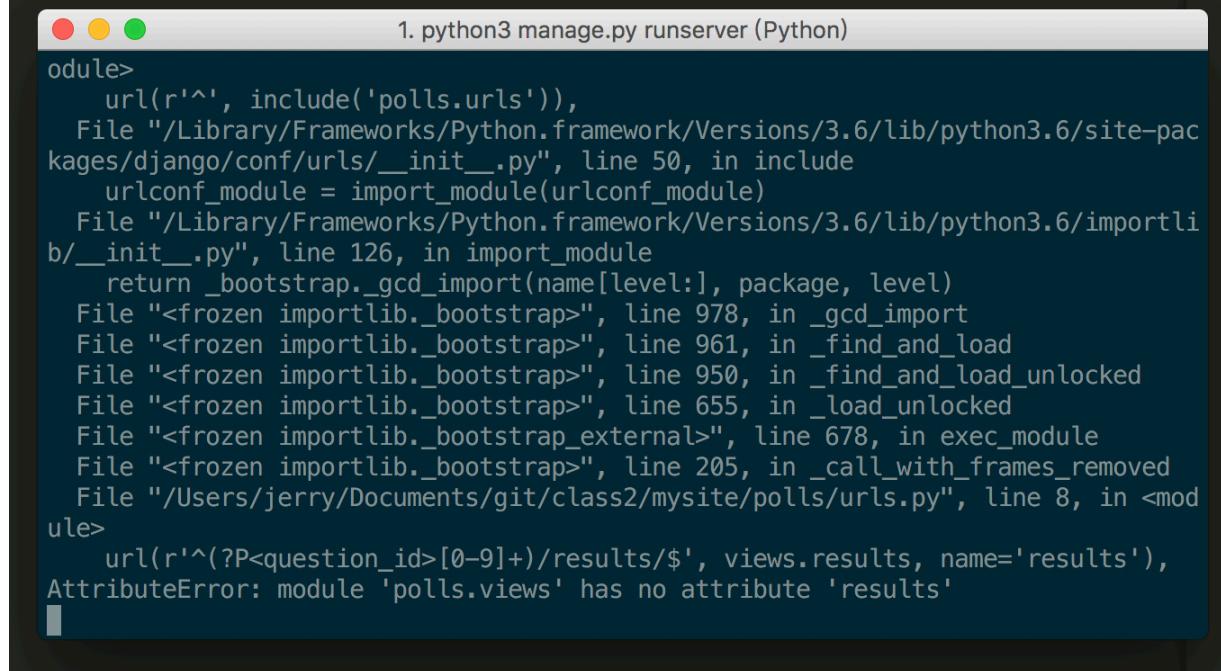
```
6 # Create your views here.
7 def index(request):
8     return HttpResponse("Hello, Polls. 你好 投票系統")
9
10 def detail(request, question_id):
11     return HttpResponse("You're looking at question %s." %
```



The screenshot shows a terminal window with a dark background and light-colored text. At the top, it says "1. python3 manage.py runserver (Python)". Below that is a detailed stack trace of a Python program. The stack trace starts with "kages/django/conf/urls/_init_.py", line 50, in include, and continues through various Python library files like "importlib._bootstrap", "importlib._bootstrap_external", and "importlib._bootstrap_external". It then points to "File "/Users/jerry/Documents/git/class2/mysite/polls/urls.py", line 3, in <module>". Inside this module, it shows the code "from . import views" and "File "/Users/jerry/Documents/git/class2/mysite/polls/views.py", line 11". The error message "return HttpResponse("You're looking at question %s." % question_id)" is shown with an upward arrow pointing to the final closing parenthesis. At the bottom of the terminal window, the error message "IndentationError: expected an indented block" is displayed.

常見錯誤

```
6 # Create your views here.
7 def index(request):
8     return HttpResponse("Hello, Polls. 你好 投票系統")
9
10 def detail(request, question_id):
11     return HttpResponse("You're looking at ques
```



The screenshot shows a terminal window titled "1. python3 manage.py runserver (Python)". The window displays a stack trace of a Python exception. The error message at the bottom reads:

```
AttributeError: module 'polls.views' has no attribute 'results'
```

```
1 from django.conf.urls import url
2
3 from . import views
4
5 urlpatterns = [
6     url(r'^$', views.index, name='index'),
7     url(r'^(?P<question_id>[0-9]+)/$', views.detail, name='detail'),
8     url(r'^(?P<question_id>[0-9]+)/results/$', views.results, name='results'),
9     url(r'^(?P<question_id>[0-9]+)/vote/$', views.vote, name='vote'),
10 ]
```

http://127.0.0.1:8000/polls/22/

127.0.0.1:8000/polls/22/

You're looking at question 22.

```
1 from django.conf.urls import url
2
3 from . import views
4
5 urlpatterns = [
6     url(r'^$', views.index, name='index'),
7     url(r'^(?P<question_id>[0-9]+)/$', views.detail, name='detail'),
8     url(r'^(?P<question_id>[0-9]+)/results/$', views.results, name='results'),
9     url(r'^(?P<question_id>[0-9]+)/vote/$', views.vote, name='vote'),
10 ]
```

<http://127.0.0.1:8000/polls/1/results/>

127.0.0.1:8000/polls/1/results/

You're looking at the results of question 1.

```
1 from django.conf.urls import url
2
3 from . import views
4
5 urlpatterns = [
6     url(r'^$', views.index, name='index'),
7     url(r'^(?P<question_id>[0-9]+)/$', views.detail, name='detail'),
8     url(r'^(?P<question_id>[0-9]+)/results/$', views.results, name='results'),
9     url(r'^(?P<question_id>[0-9]+)/vote/$', views.vote, name='vote'),
10 ]
```

<http://127.0.0.1:8000/polls/1/results/>

127.0.0.1:8000/polls/1/vote/

You're voting on question 1.

Write views that actually do something

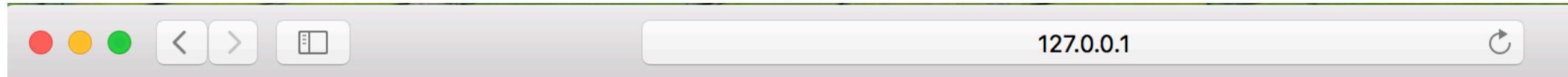
- Each view is responsible for doing one of two things: returning an [HttpResponse](#) object containing the content for the requested page, or raising an exception such as [Http404](#).

polls/views.py

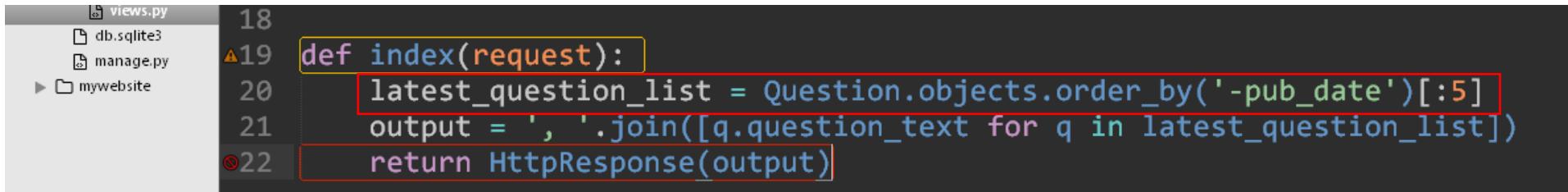
```
OPEN FILES
views.py
urls.py
FOLDERS
Django_new
chapter1
mysite
mysite
__pycache__
_init_.py
settings.py
urls.py
wsgi.py
polls
__pycache__
migrations
templates
polls
detail.html
index.html
_init_.py
admin.py
apps.py
models.py
tests.py
urls.py
views.py
db.sqlite3
manage.py
mywebsite
views.py
* views.py
  urls.py

1 from django.http import HttpResponseRedirect
2
3 # Create your views here.
4 from .models import Question
5
6 def index(request):
7     return HttpResponseRedirect("Hello!")
8
9 def detail(request, question_id):
10    return HttpResponseRedirect("You're looking at question %s." % question_id)
11
12 def results(request, question_id):
13    response = "You're looking at the results of question %s."
14    return HttpResponseRedirect(response % question_id)
15
16 def vote(request, question_id):
17    return HttpResponseRedirect("You're voting on question %s." % question_id)
18
19 def index(request):
20    latest_question_list = Question.objects.order_by('-pub_date')[:5]
21    output = ', '.join([q.question_text for q in latest_question_list])
22    return HttpResponseRedirect(output)
```

`http://127.0.0.1:8000/jerry/5/vote/`



Python Shell



A screenshot of a code editor showing a file named 'views.py'. The code defines a function 'index' that retrieves the five latest questions from the database and returns them as an HttpResponse. The code is highlighted with syntax coloring, and specific lines are outlined with red boxes.

```
views.py
18
19 def index(request):
20     latest_question_list = Question.objects.order_by('-pub_date')[:5]
21     output = ', '.join([q.question_text for q in latest_question_list])
22     return HttpResponseRedirect(output)
```

```
In [1]: from polls.models import Question, Choice
```

```
In [2]: Question.objects.all()
```

```
Out[2]: <QuerySet [<Question: 這是問題1>, <Question: 問題2>]>
```

```
In [3]: Question.objects.order_by('-pub_date')[:5]
```

```
Out[3]: <QuerySet [<Question: 問題2>, <Question: 這是問題1>]>
```

```
In [4]: Question.objects.order_by('pub_date')[:5]
```

```
Out[4]: <QuerySet [<Question: 這是問題1>, <Question: 問題2>]>
```

```
In [5]: 
```

Python Shell

```
views.py
18
19 def index(request):
20     latest_question_list = Question.objects.order_by('-pub_date')[:5]
21     output = ', '.join([q.question_text for q in latest_question_list])
22     return HttpResponse(output)
```

```
In [30]: x = ['my', 'name', 'is', 'bob']
```

```
In [31]: ', '.join(x)
```

```
Out[31]: 'my, name, is, bob'
```

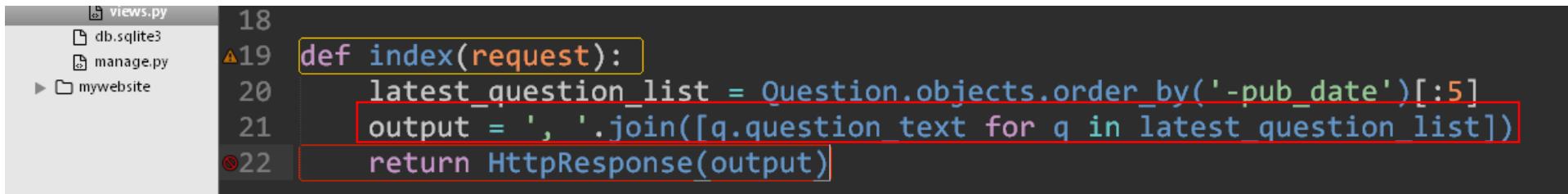
```
In [32]:
```

```
In [41]: for i in x:
...:     print(i)
...:
my
name
is
bob
```

```
In [44]: x_list = [i for i in x]
```

```
In [45]: print(x_list)
['my', 'name', 'is', 'bob']
```

Python Shell



A screenshot of a code editor showing a file named `views.py`. The code defines a function `index` that retrieves the five latest questions from the database and returns them as a string. The code is highlighted with syntax coloring.

```
views.py
18
19 def index(request):
20     latest_question_list = Question.objects.order_by('-pub_date')[:5]
21     output = ', '.join([q.question_text for q in latest_question_list])
22     return HttpResponseRedirect(output)
```

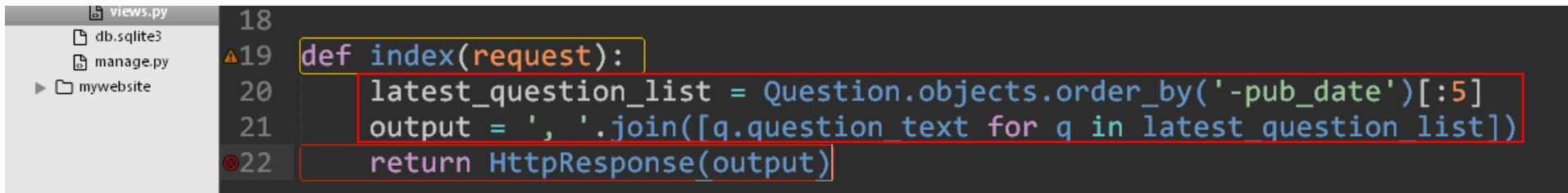
```
In [46]: ', '.join( [i for i in x] )
Out[46]: 'my, name, is, bob'

In [47]: join_list = ', '.join( [i for i in x] )

In [48]: type(join_list)
Out[48]: str

In [49]: 
```

Python Shell



```
views.py
18
19 def index(request):
20     latest_question_list = Question.objects.order_by('-pub_date')[:5]
21     output = ', '.join([q.question_text for q in latest_question_list])
22     return HttpResponseRedirect(output)
```

```
In [63]: latest_question_list = Question.objects.order_by('-pub_date')[:5]

In [64]: ', '.join([q.question_text for q in latest_question_list])
Out[64]: '問題2, 這是問題1'

In [65]: [q.question_text for q in latest_question_list]
Out[65]: ['問題2', '這是問題1']

In [66]: output = ', '.join([q.question_text for q in latest_question_list])

In [67]: HttpResponseRedirect(output)
Out[67]: <HttpResponse status_code=200, "text/html; charset=utf-8">

In [68]: print(output)
問題2, 這是問題1
```

把問題標題，轉成字串輸出。

```
In [69]:
```

常見錯誤

NameError at /polls/

name 'Question' is not defined

Request Method: GET

Request URL: http://127.0.0.1:8000/polls/

Django Version: 1.11rc1

Exception Type: NameError

Exception Value: name 'Question' is not defined

Exception Location: /Users/jerry/Documents/git/class2/Chapter1_test/polls/views.py in index, line 6

Python Executable: /usr/local/bin/python3

Python Version: 3.6.0

Python Path: ['/Users/jerry/Documents/git/class2/Chapter1_test',
 '/Library/Frameworks/Python.framework/Versions/3.6/lib/python36.zip',
 '/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6',
 '/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/lib-dynload',
 '/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/site-packages']

Server time: Wed, 10 May 2017 23:48:46 +000



```
models.py * views.py * urls.py * settings.py * admin.py *  
1 from django.shortcuts import render  
2 from django.http import HttpResponseRedirect  
3 from .models import Question  
4 # Create your views here.  
5 def index(request):  
6     latest_question_list = Question.objects.order_by('-pub_date')[:5]  
7     output = ', '.join([q.question_text for q in latest_question_list])  
8     return HttpResponseRedirect(output)  
9
```

✓ The question "五月天厲害嗎?" was added successfully.

Select question to change

ADD QUESTION +

Action: ----- ▾ Go 0 of 3 selected

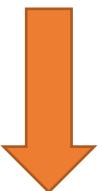
QUESTION

五月天厲害嗎?

What's up?

What's up

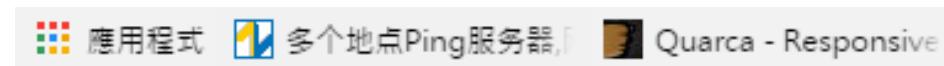
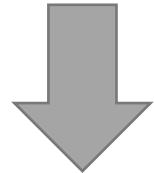
3 questions



五月天厲害嗎?, What's up?, What's up



如果要編輯頁面的內容，要如何做？
第一步驟 加上Templates
第二步驟 加上Render



- 五月天厲害嗎?
- What's up?
- What's up

polls/templates/polls/index.html

The screenshot shows a code editor interface with a dark theme. On the left is a sidebar titled "OPEN FILES" and "FOLDERS". The "OPEN FILES" section lists "views.py", "index.html", and "urls.py". The "FOLDERS" section shows a project structure: "Django_new" contains "chapter1", "mysite" (which has "mysite" and "polls" subfolders), and "polls" (which has "templates" and "polls" subfolders). Inside "polls/templates/polls", there are files "detail.html" and "index.html". The "index.html" file is currently selected and displayed in the main editor area. Its content is:

```
1  {% if latest_question_list %}  
2      <ul>  
3          {% for question in latest_question_list %}  
4              <li><a href="/polls/{{ question.id }}/">{{ question.question_text }}</a>  
5                  </li>  
6          {% endfor %}  
7      </ul>  
8  {% else %}  
9      <p>No polls are available.</p>  
10  {% endif %}
```

polls/views.py

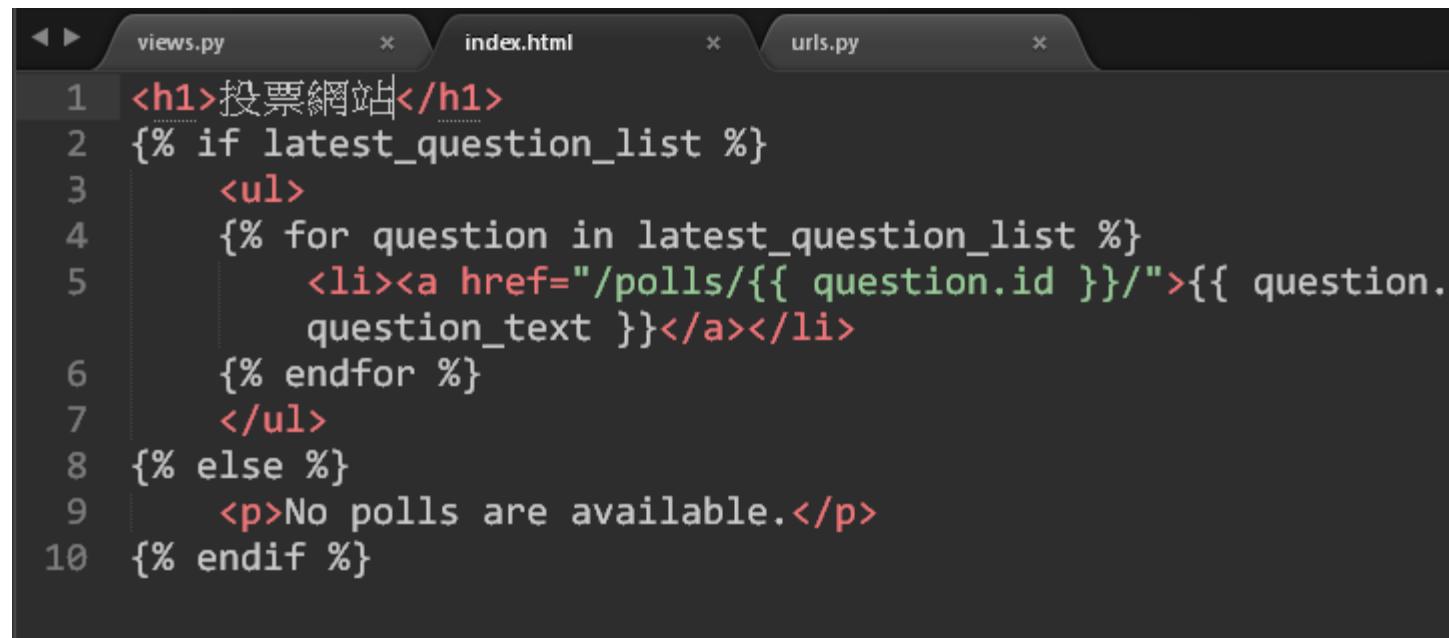
dsjerry2017@gmail.com

```
OPEN FILES
  x views.py
  x index.html
  x urls.py
FOLDERS
  ▼ Django_new
    ▶ chapter1
    ▶ mysite
      ▶ mysite
        □ __pycache__
          □ __init__.py
          □ settings.py
          □ urls.py
          □ wsgi.py
      ▶ polls
        ▶ __pycache__
        ▶ migrations
        ▶ templates
          ▶ polls
            □ detail.html
            □ index.html
          □ __init__.py
          □ admin.py
          □ apps.py
          □ models.py
          □ tests.py
          □ urls.py
          □ views.py
        □ db.sqlite3
        □ manage.py
    ▶ mywebsite

views.py
  ◀ ▶ index.html x index.html x urls.py x
  1 from django.http import HttpResponseRedirect
  2 from django.template import loader
  3
  4 # Create your views here.
  5 from .models import Question
  6
  7 def index(request):
  8     return HttpResponseRedirect("Hello!")
  9
  10 def detail(request, question_id):
  11     return HttpResponseRedirect("You're looking at question %s." % question_id)
  12
  13 def results(request, question_id):
  14     response = "You're looking at the results of question %s."
  15     return HttpResponseRedirect(response % question_id)
  16
  17 def vote(request, question_id):
  18     return HttpResponseRedirect("You're voting on question %s." % question_id)
  19
  20 def index(request):
  21     latest_question_list = Question.objects.order_by('-pub_date')[:5]
  22     template = loader.get_template('polls/index.html')
  23     context = {
  24         'latest_question_list': latest_question_list,
  25     }
  26     return HttpResponseRedirect(template.render(context, request))
```

投票網站

- 五月天厲害嗎?
- What's up?
- What's up



The screenshot shows a code editor with three tabs: 'views.py', 'index.html', and 'urls.py'. The 'index.html' tab is active and displays the following Python template code:

```
1 <h1>投票網站</h1>
2 {% if latest_question_list %}
3     <ul>
4         {% for question in latest_question_list %}
5             <li><a href="/polls/{{ question.id }}/">{{ question.
6                 question_text }}</a></li>
7         {% endfor %}
8     </ul>
9     {% else %}
10        <p>No polls are available.</p>
11    {% endif %}
```

A shortcut: render()

```
21 def index(request):
22     latest_question_list = Question.objects.order_by('-pub_date')[:5]
23     template = loader.get_template('polls/index.html')
24     context = {
25         'latest_question_list': latest_question_list,
26     }
27     return HttpResponseRedirect(template.render(context, request))
```

設定Templates的位置



```
28
29 def index(request):
30     latest_question_list = Question.objects.order_by('-pub_date')[:5]
31     context = {'latest_question_list': latest_question_list}
32     return render(request, 'polls/index.html', context)
```

A shortcut: render()

The screenshot shows a code editor with the following file structure:

- OPEN FILES:
 - views.py
 - index.html
 - urls.py
- FOLDERS:
 - Django_new
 - chapter1
 - mysite
 - mysite
 - __pycache__
 - __init__.py
 - settings.py
 - urls.py
 - wsgi.py
 - polls
 - __pycache__
 - migrations
 - templates
 - polls
 - detail.html
 - index.html
 - __init__.py
 - admin.py
 - apps.py
 - models.py
 - tests.py
 - urls.py
 - views.py
 - db.sqlite3
 - manage.py
 - mywebsite

The code in views.py is as follows, with specific sections highlighted:

```
from django.template import loader
from django.shortcuts import render

# Create your views here.
from .models import Question

def index(request):
    return HttpResponse("Hello!")

def detail(request, question_id):
    return HttpResponse("You're looking at question %s." % question_id)

def results(request, question_id):
    response = "You're looking at the results of question %s."
    return HttpResponse(response % question_id)

def vote(request, question_id):
    return HttpResponse("You're voting on question %s." % question_id)

def index(request):
    latest_question_list = Question.objects.order_by('-pub_date')[:5]
    context = {'latest_question_list': latest_question_list}
    return render(request, 'polls/index.html', context)
```

The code editor highlights several sections with yellow boxes:

 - Line 2: `from django.template import loader`
 - Line 3: `from django.shortcuts import render`
 - Line 8: `def index(request):`
 - Line 9: `return HttpResponse("Hello!")`
 - Line 11: `def detail(request, question_id):`
 - Line 12: `return HttpResponse("You're looking at question %s." % question_id)`
 - Line 14: `def results(request, question_id):`
 - Line 15: `response = "You're looking at the results of question %s."`
 - Line 16: `return HttpResponse(response % question_id)`
 - Line 18: `def vote(request, question_id):`
 - Line 19: `return HttpResponse("You're voting on question %s." % question_id)`
 - Line 22: `def index(request):`
 - Line 23: `latest_question_list = Question.objects.order_by('-pub_date')[:5]`
 - Line 24: `context = {'latest_question_list': latest_question_list}`
 - Line 25: `return render(request, 'polls/index.html', context)`

A large red box surrounds the entire final `index` function definition from line 22 to line 25.

A shortcut: get_object_or_404()

dsjerry2017@gmail.com

The screenshot shows a code editor with a dark theme. On the left is a sidebar titled 'OPEN FILES' containing 'views.py'. Below it is a 'FOLDERS' tree with 'Django_new' expanded, showing 'chapter1', 'mysite' (expanded), 'mysite' (under 'mysite'), '_pycache_', '__init__.py', and 'settings.py'. The main window shows the code for 'views.py':

```
1 from django.http import HttpResponseRedirect
2 from django.template import loader
3 from django.shortcuts import get_object_or_404, render
4 from django.http import Http404
5
6 # Create your views here.
7 from .models import Question
```

Line 3, 'from django.shortcuts import get_object_or_404, render', is highlighted with a red box.

Page not found (404)

Request Method: GET

Request URL: http://127.0.0.1:8000/polls/11/

Raised by: polls.views.detail

No Question matches the given query.

You're seeing this error because you have `DEBUG = True` in your Django settings file. Change that to `False`, and Django will display a standard 404 page.

The screenshot shows a code editor with a dark theme. On the left is a sidebar titled 'OPEN FILES' containing 'admin.py', 'apps.py', 'models.py', 'tests.py', 'urls.py', 'views.py' (selected), 'db.sqlite3', and 'manage.py'. Below it is a 'FOLDERS' tree with 'mywebsite' expanded. The main window shows the code for 'views.py':

```
10     return HttpResponseRedirect("You're voting on question %s." % question_id)
11
12
13 def index(request):
14     latest_question_list = Question.objects.order_by('-pub_date')[:5]
15     context = {'latest_question_list': latest_question_list}
16     return render(request, 'polls/index.html', context)
17
18 def detail(request, question_id):
19     question = get_object_or_404(Question, pk=question_id)
20     return render(request, 'polls/detail.html', {'question': question})
```

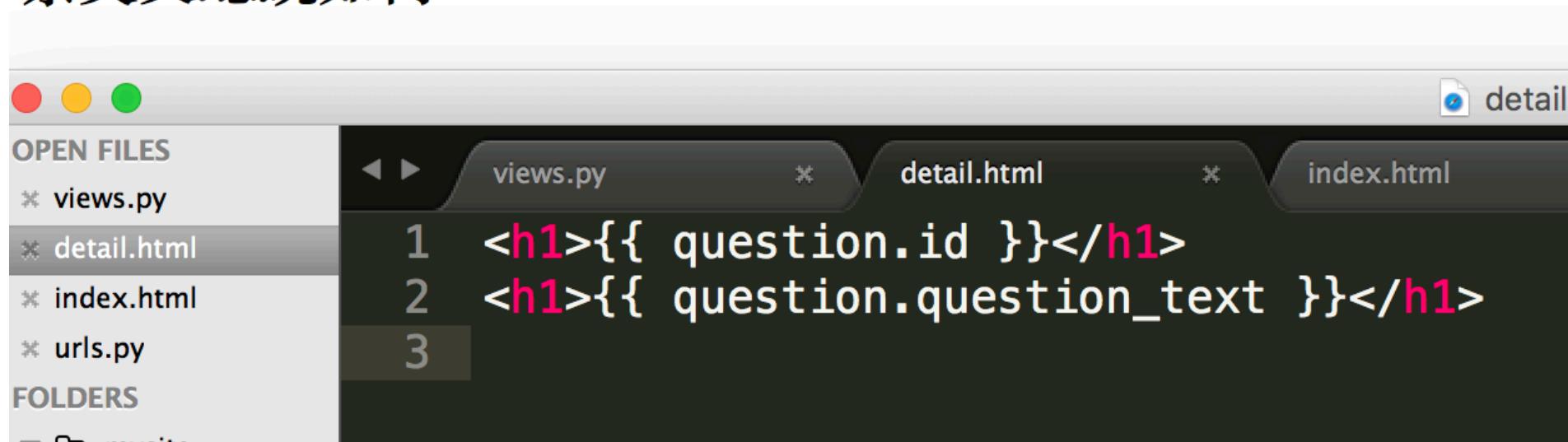
Line 18, 'def detail(request, question_id):', is highlighted with a red box.

務必記得加上detail.html

```
polls/templates/polls/detail.html  
{{ question }}
```

3

蔡英文總統如何？



The screenshot shows a code editor interface with the following details:

- OPEN FILES:** A sidebar on the left lists five files: views.py, detail.html, index.html, and urls.py.
- Tab Bar:** At the top, there are tabs for views.py, detail.html, and index.html. The detail.html tab is currently active.
- Code View:** The main area displays the contents of the detail.html file. The code is as follows:

```
<h1>{{ question.id }}</h1>
<h1>{{ question.question_text }}</h1>
```

A number "3" is visible at the bottom left of the code area.

Namespacing URL names

dsjerry2017@gmail.com

- How does one make it so that Django knows which app view to create for a url when using the {% url %} template tag? (方便識別自己用的APP)

polls/urls.py

```
from django.conf.urls import url

from . import views

app_name = 'polls'
urlpatterns = [
    url(r'^$', views.index, name='index'),
    url(r'^(?P<question_id>[0-9]+)/$', views.detail, name='detail'),
    url(r'^(?P<question_id>[0-9]+)/results/$', views.results, name='results'),
    url(r'^(?P<question_id>[0-9]+)/vote/$', views.vote, name='vote'),
]
```

polls/templates/polls/index.html

```
<li><a href="{% url 'polls:detail' question.id %}">{{ question.question_text }}</a></li>
```

Part 4: Forms and generic views

HTML Forms

- **action** :The **action** attribute defines the action to be performed when the form is submitted.
Normally, the form data is sent to a web page on the server when the user clicks on the submit button.

```
1 <h1>{{ question.question_text }}</h1>
2
3 {% if error_message %}<p><strong>{{ error_message }}</strong></p>{% endif %}
4 |
5 <form action="{% url 'polls:vote' question.id %}" method="post">
6   {% csrf_token %}
7   {% for choice in question.choice_set.all %}
8     <input type="radio" name="choice" id="choice{{ forloop.counter }}" value="{{ choice.id }}" />
9     <label for="choice{{ forloop.counter }}">{{ choice.choice_text }}</label><br />
10  {% endfor %}
11  <input type="submit" value="Vote" />
12 </form>
```

HTML Forms

- The **method** attribute specifies the HTTP method (**GET** or **POST**) to be used when submitting the form data

```
1 <h1>{{ question.question_text }}</h1>
2
3 {% if error_message %}<p><strong>{{ error_message }}</strong></p>{% endif %}
4 |
5 <form action="{% url 'polls:vote' question.id %}" method="post">
6   {% csrf_token %}
7   {% for choice in question.choice_set.all %}
8     <input type="radio" name="choice" id="choice{{ forloop.counter }}" value="{{ choice.id }}" />
9     <label for="choice{{ forloop.counter }}">{{ choice.choice_text }}</label><br />
10  {% endfor %}
11  <input type="submit" value="Vote" />
12 </form>
```

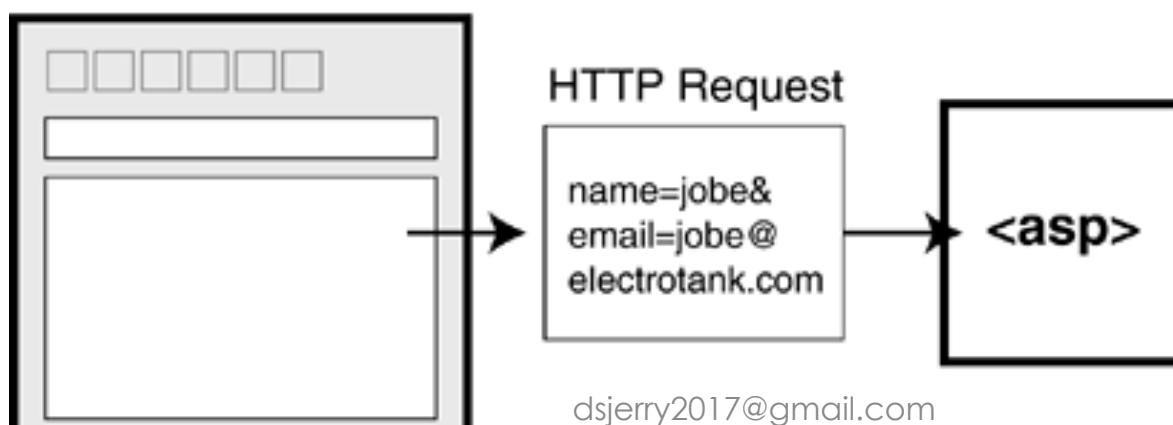
Using GET

http://www.somedomain.com/register.asp?name=jobe&email=jobe@electrotank.com



Using POST

http://www.somedomain.com/register.asp



HTML Forms (detail.html)

- `<input type="radio">` defines a **radio button**.
- Radio buttons let a user select ONLY ONE of a limited number of choices:

```
1 <h1>{{ question.question_text }}</h1>
2
3 {% if error_message %}<p><strong>{{ error_message }}</strong></p>{% endif %}
4
5 <form action="{% url 'polls:vote' question.id %}" method="post">
6 {% csrf_token %}
7 {% for choice in question.choice_set.all %}
8     <input type="radio" name="choice" id="choice{{ forloop.counter }}" value="{{ choice.id }}" />
9     <label for="choice{{ forloop.counter }}">{{ choice.choice_text }}</label><br />
10 {% endfor %}
11 <input type="submit" value="Vote" />
12 </form>
```

```
<!DOCTYPE html>
<html>
<body>

<form action="/action_page.php">
    <input type="radio" name="gender" value="male"> Male<br>
    <input type="radio" name="gender" value="female"> Female<br>
    <input type="radio" name="gender" value="other"> Other<br><br>
    <input type="submit">
</form>
```

- Male
- Female
- Other

提交

```
1 from django.http import HttpResponseRedirect, HttpResponse
2 from django.template import loader
3 from django.shortcuts import get_object_or_404, render
4 from django.http import Http404
5 from django.urls import reverse
6
```

```
33 def vote(request, question_id):
34     question = get_object_or_404(Question, pk=question_id)
35     try:
36         selected_choice = question.choice_set.get(pk=request.POST['choice'])
37     except (KeyError, Choice.DoesNotExist):
38         # Redisplay the question voting form.
39         return render(request, 'polls/detail.html', {
40             'question': question,
41             'error_message': "You didn't select a choice.",
42         })
43     else:
44         selected_choice.votes += 1
45         selected_choice.save()
46         # Always return an HttpResponseRedirect after successfully dealing
47         # with POST data. This prevents data from being posted twice if a
48         # user hits the Back button.
49         return HttpResponseRedirect(reverse('polls:results', args=(question.id,)))
```

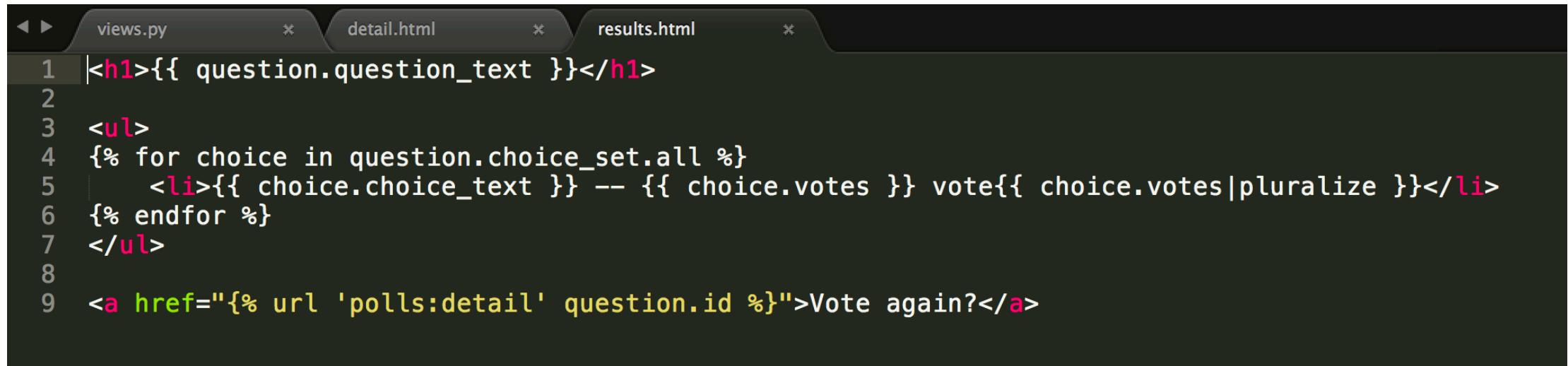
Add Options

加入選項

```
1 from django.contrib import admin
2
3 # Register your models here.
4 from .models import Choice, Question
5
6 class ChoiceInline(admin.TabularInline):
7     model = Choice
8     extra = 3
9
10 class QuestionAdmin(admin.ModelAdmin):
11     fieldsets = [
12         (None, {'fields': ['question_text']}), 
13         ('Date information', {'fields': ['pub_date'], 'classes': ['collapse']}), 
14     ]
15     inlines = [ChoiceInline]
16     list_display = ('question_text', 'pub_date')
17     list_filter = ['pub_date']
18     search_fields = ['question_text']
19
20
21 admin.site.register(Question, QuestionAdmin)
```

Add Results

Results.html



The screenshot shows a code editor with a dark theme. The tab bar at the top has four tabs: 'views.py', 'detail.html', 'results.html' (which is the active tab), and another unnamed tab. The code in the 'results.html' tab is as follows:

```
1 <h1>{{ question.question_text }}</h1>
2
3 <ul>
4 {% for choice in question.choice_set.all %}
5   <li>{{ choice.choice_text }} -- {{ choice.votes }} vote{{ choice.votes|pluralize }}</li>
6 {% endfor %}
7 </ul>
8
9 <a href="{% url 'polls:detail' question.id %}">Vote again?</a>
```

Views.py Check

```
1 from django.shortcuts import render
2 from django.http import HttpResponseRedirect, HttpResponse
3 from django.urls import reverse
4 from django.shortcuts import get_object_or_404, render
5 from .models import Choice, Question
6
7 # Create your views here.
8 def index(request):
9     latest_question_list = Question.objects.order_by('-pub_date')[:5]
10    context = {'latest_question_list': latest_question_list}
11    return render(request, 'jerry/index.html', context)
12
13 def detail(request, question_id):
14     question = get_object_or_404(Question, pk=question_id)
15     return render(request, 'jerry/detail.html', {'question': question})
16
17 def results(request, question_id):
18     question = get_object_or_404(Question, pk=question_id)
19     return render(request, 'jerry/results.html', {'question': question})
20
21 def vote(request, question_id):
22     question = get_object_or_404(Question, pk=question_id)
23     try:
24         selected_choice = question.choice_set.get(pk=request.POST['choice'])
25     except (KeyError, Choice.DoesNotExist):
26         # Redisplay the question voting form.
27         return render(request, 'jerry/detail.html', {
28             'question': question,
29             'error_message': "You didn't select a choice.",
30         })
31     else:
32         selected_choice.votes += 1
33         selected_choice.save()
34         # Always return an HttpResponseRedirect after successfully dealing
35         # with POST data. This prevents data from being posted twice if a
36         # user hits the Back button.
37         return HttpResponseRedirect(reverse('jerry:results', args=(question.id,)))
```

誰是校花

A

B

C

Deploy Heroku

維基百科，自由的百科全書

Heroku是一個支援多種程式語言的雲平台即服務。在2010年被Salesforce.com收購。Heroku作為最開始的雲平台之一^[1]，從2007年6月起開發，當時它僅支援Ruby，但後來增加了對Java、Node.js、Scala、Clojure、Python以及（未記錄在正式檔案上）PHP和Perl的支援。基礎作業系統是Debian，在最新的堆疊則是基於Debian的Ubuntu^[2]。

目錄 [隱藏]

- 1 歷史
- 2 競爭對手
- 3 參考資料
- 4 外部連結

歷史 [編輯]

James Lindenbaum、Adam Wiggins和Orion Henry建立了Heroku，用於支援相容Rack的專案。^[3]2009年10月，Byron Sebastian作為CEO加入了Heroku。^[4]2010年12月8日，Heroku成Salesforce.com完全控股的子公司。在2011年6月12日，Ruby的首席設計師松本行弘加盟該公司，擔任Ruby的首席架構師。^[5]當月，Heroku加入了Node.js和Clojure的支援。2011年9月15日，Heroku和Facebook合作。^[6]現在，除了基礎的PostgreSQL^[7]之外，Heroku還支援Cloudant、Couchbase Server、MongoDB和Redis^[8]，既作為平台的一部分，又作為一個獨立的服務。^[9]在Heroku伺服器上執行的應用程式使用Heroku的DNS伺服器連線至應用程式的域名（通常為「應用程式名.herokuapp.com」）。每個應用程式容器（稱為dyno）跨越由多個伺服器組成的「dyno grid」。Heroku的Git伺服器處理來自授權用戶的應用程式庫推播請求。^[10]

2012年北美雷雨導致很多Heroku代管的應用離機，服務持續中斷近24小時。^[11]

Heroku, Inc.



公司類型 子公司

成立 2007年

創辦人 James Lindenbaum, Adam Wiggins, Orion Henry

代表人物 Tod Nielsen, CEO

總部 加利福尼亞州舊金山

產業 雲端平台即服務

母公司 Salesforce.com

網站 heroku.com



Free

Ideal for experimenting with cloud applications in a limited sandbox.

CORE PLATFORM FEATURES

SLEEPS AFTER 30 MINS OF INACTIVITY

USES AN ACCOUNT-BASED POOL
OF FREE DYNOS

CUSTOM DOMAINS

512 MB RAM | 1 web/1 worker

Free



Hobby

Perfect for small scale personal projects and hobby apps.

CORE PLATFORM FEATURES

NEVER SLEEPS

FREE SSL & AUTOMATED CERTIFICATE
MANAGEMENT FOR CUSTOM DOMAINS

APPLICATION METRICS

MULTIPLE WORKERS FOR MORE
POWERFUL APPS

512 MB RAM | 10 Process Types

\$7 per dyno/month
prorated to the second



Standard

1X 2X

Enhanced visibility, performance, and availability for powering your professional applications.

ALL HOBBY FEATURES +

SIMPLE HORIZONTAL SCALABILITY

THRESHOLD ALERTS

PREBOOT

512MB OR 1GB RAM

∞ Process Types



\$25 - \$500 per dyno/month
prorated to the second



Performance

M L

Superior performance when it's most critical for your super scale, high traffic apps.

ALL STANDARD FEATURES +

MIX WITH STANDARD 1X, 2X DYNOS

DEDICATED

AUTOSCALING

2.5GB OR 14GB RAM

Set up

- heroku create
- git push heroku master
- heroku ps:scale web=1
- heroku open

<https://devcenter.heroku.com/articles/getting-started-with-python#set-up>

Add urls.py

```
urls.py  
*  
1 from django.conf.urls import include, url  
2  
3 from django.contrib import admin  
4 admin.autodiscover()  
5  
6 import hello.views  
7  
8 # Examples:  
9 # url(r'^$', 'gettingstarted.views.home', name='home'),  
10 # url(r'^blog/', include('blog.urls')),  
11  
12 urlpatterns = [  
13     url(r'^$', hello.views.index, name='index'),  
14     url(r'^db', hello.views.db, name='db'),  
15     url(r'^admin/', include(admin.site.urls)),  
16     url(r'^jerry/', include('jerry.urls')),  
17  
18 ]  
19
```

Check files

 app.json	 __init__.py
 gettingstarted	 __init__.pyc
 hello	 __pycache__
 jerry	 admin.py
 manage.py	 admin.pyc
 Pipfile	 apps.py
 Pipfile.lock	 migrations
 Procfile	 models.py
 Procfile.windows	 models.pyc
 README.md	 templates
	 tests.py
	 urls.py
	 urls.pyc
	 views.py
	 views.pyc

Add app

```
28 DEBUG = True
29
30 TEST_RUNNER = 'gettingstarted.heroku_test_runner.HerokuDiscoverRunner'
31
32
33 # Application definition
34
35 INSTALLED_APPS = (
36     'django.contrib.admin',
37     'django.contrib.auth',
38     'django.contrib.contenttypes',
39     'django.contrib.sessions',
40     'django.contrib.messages',
41     'django.contrib.staticfiles',
42     'hello',
43     'jerry',
44 )
45
```

```
git add .  
git commit -m "new"  
git push  
Heroku run python3 manage.py migrate
```



- 選三
- 選二
- 選一

[Vote](#)

<https://agile-everglades-90812.herokuapp.com/jerry/1>

CMS APP - Initial

First Step

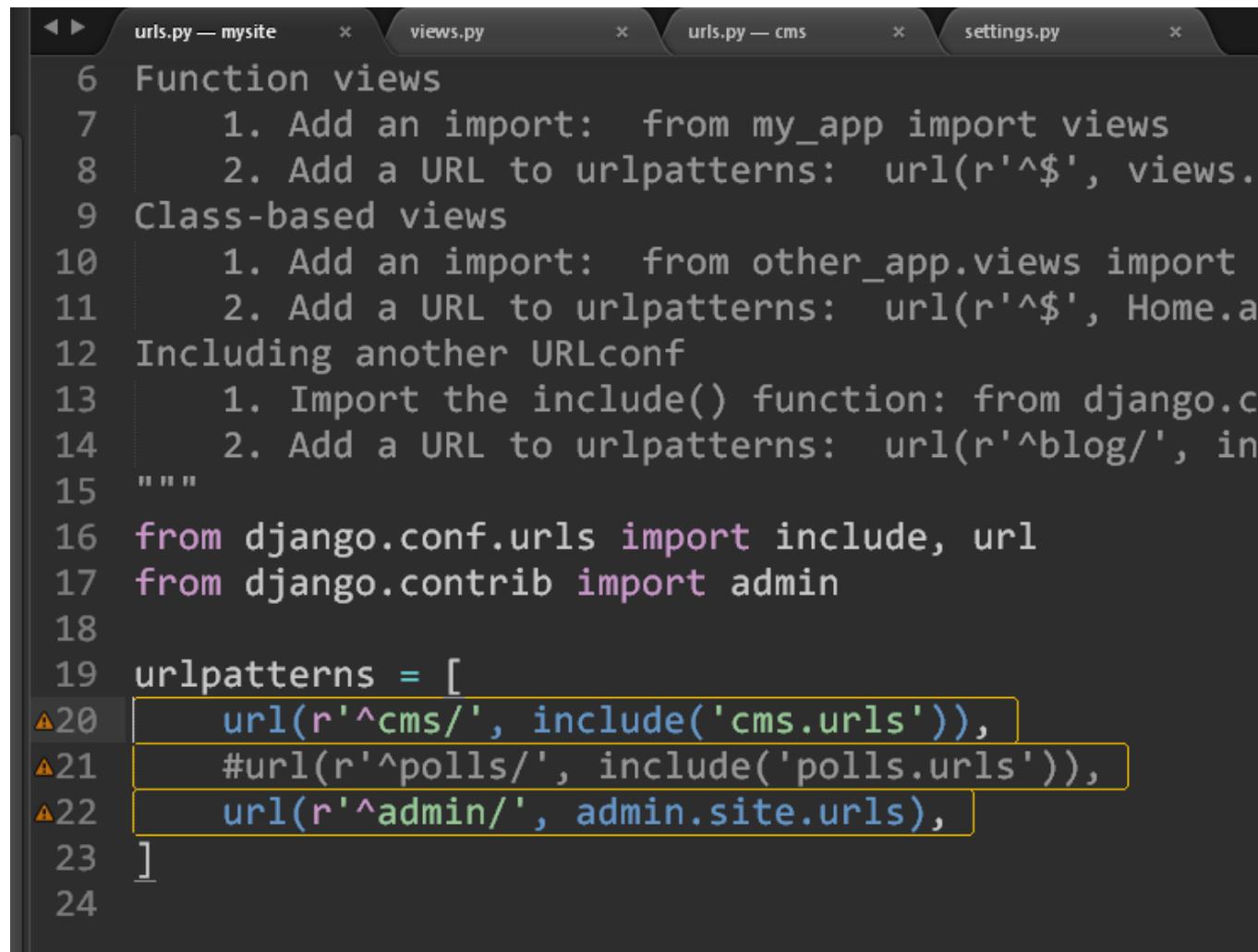
- mysite
 - settings.py
 - urls.py
- cms
 - urls.py
 - views.py

mysite/settings.py

```
# Application definition

INSTALLED_APPS = [
    'cms.apps.CmsConfig',
    'polls.apps.PollsConfig',
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
```

mysite/urls.py



```
urls.py — mysite      x  views.py      x  urls.py — cms      x  settings.py      x
6 Function views
7     1. Add an import: from my_app import views
8         2. Add a URL to urlpatterns: url(r'^$', views.h
9 Class-based views
10    1. Add an import: from other_app.views import H
11        2. Add a URL to urlpatterns: url(r'^$', Home.as
12 Including another URLconf
13    1. Import the include() function: from django.co
14        2. Add a URL to urlpatterns: url(r'^blog/', inc
15 """
16 from django.conf.urls import include, url
17 from django.contrib import admin
18
19 urlpatterns = [
20     url(r'^cms/', include('cms.urls')),
21     #url(r'^polls/', include('polls.urls')),
22     url(r'^admin/', admin.site.urls),
23 ]
24
```

cms/urls.py

```
1 from django.conf.urls import url  
2  
3 from . import views  
4  
5 urlpatterns = [  
6     url(r'^$', views.index, name='index'),  
7 ]
```

cms/views.py

The image shows a code editor interface with a sidebar displaying the project's directory structure. The main pane shows the content of the `views.py` file.

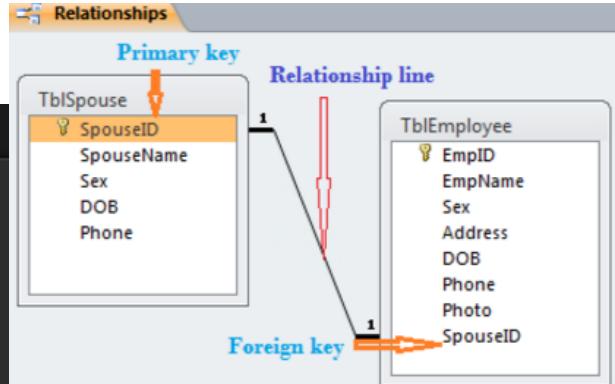
```
1 from django.shortcuts import render
2 from django.http import HttpResponseRedirect
3 # Create your views here.
4
5 def index(request):
6     return HttpResponseRedirect("Hello, world.")
```

The sidebar shows the following directory structure:

- chapter1
- Chapter2
- Chapter3
- mysite
 - cms
 - __pycache__
 - migrations
 - __init__.py
 - admin.py
 - apps.py
 - models.py
 - tests.py
 - urls.py
 - views.py
 - __pycache__
 - __init__.py
 - settings.py
 - urls.py
 - wsgi.py

CMS APP - Model

- `python manage.py migrate`



The diagram illustrates a relationship between two tables: **TblSpouse** and **TblEmployee**. **TblSpouse** is defined as the primary key, indicated by a red arrow pointing to its **SpouseID** field. A relationship line connects **TblSpouse** to **TblEmployee**, with a red arrow labeled "Foreign key" pointing from **TblEmployee** back to **TblSpouse**, specifically to the **SpouseID** field. Both tables share the same structure: **SpouseName**, **Sex**, **DOB**, and **Phone**.

```

models.py
1 from django.db import models
2
3 # Create your models here.
4 class Restaurant(models.Model):
5     name = models.CharField(max_length=20)
6     phone_number = models.CharField(max_length=15)
7     address = models.CharField(max_length=50, blank=True)
8
9
10 class Food(models.Model):
11     name = models.CharField(max_length=20) 小數的最大位數為0
12     price = models.DecimalField(max_digits=3, decimal_places=0)
13     comment = models.CharField(max_length=50, blank=True)
14     is_spicy = models.BooleanField(default=False)
15     restaurant = models.ForeignKey(Restaurant)

```

Activating models

- python manage.py makemigrations cms
- python manage.py migrate

```
--  
ALTER TABLE "cms_food" RENAME TO "cms_food_old";  
CREATE TABLE "cms_food" <"id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "name"  
varchar(20) NOT NULL, "price" decimal NOT NULL, "comment" varchar(50) NOT NULL  
"is_spicy" bool NOT NULL, "restaurant_id" integer NOT NULL REFERENCES "cms_res-  
taurant" <"id">;  
INSERT INTO "cms_food" <"price", "name", "id", "is_spicy", "restaurant_id", "co-  
ment> SELECT "price", "name", "id", "is_spicy", NULL, "comment" FROM "cms_food_  
old";  
DROP TABLE "cms_food_old";  
CREATE INDEX "cms_food_restaurant_id_0d7c2f2f" ON "cms_food" <"restaurant_id">;  
COMMIT;  
  
mywebsite> c:\GitHub\ Django_new\mysite>python manage.py migrate  
operations to perform:  
  Apply all migrations: admin, auth, cms, contenttypes, sessions  
running migrations:  
  Applying cms.0001_initial... OK
```

admin.py

```
1 from django.contrib import admin  
2  
3 # Register your models here.  
4 from .models import Restaurant,Food  
5  
6 admin.site.register(Food)  
7 admin.site.register(Restaurant)
```

Add food

Name:

Price:

Comment:

 Is spicy

Restaurant:

 ----- ▾[Save and add another](#)[Save and continue editing](#)[SAVE](#)

Select food to change

[ADD FOOD +](#)Action: ----- ▾ 0 of 1 selected

<input type="checkbox"/>	NAME	PRICE
<input checked="" type="checkbox"/>	辣味義大利麵	100

1 food

dsjerry2017@gmail.com

```
◀ ▶ settings.py      × admin.py      × models.py      ×
 1 from django.db import models
 2
 3 # Create your models here.
 4 class Restaurant(models.Model):
 5     name = models.CharField(max_length=20)
 6     phone_number = models.CharField(max_length=15)
 7     address = models.CharField(max_length=50, blank=True)
 8
 9     def __str__(self):
10         return self.name
11
12 class Food(models.Model):
13     name = models.CharField(max_length=20)
14     price = models.DecimalField(max_digits=3, decimal_places=0)
15     comment = models.CharField(max_length=50, blank=True)
16     is_spicy = models.BooleanField(default=False)
17     restaurant = models.ForeignKey(Restaurant)
18
19     def __str__(self):
20         return self.name
```

Change food

HISTORY

Name:

辣味義大利麵

Price:

100

Comment:

特別美食

Is spicy

Restaurant:

意素家



Delete

Save and add another

Save and continue editing

SAVE

CMS APP - Shell

Display

- python manage.py shell
- from cms.models import Restaurant, Food
- restaurants=Restaurant.objects.all()
- restaurants

Create

- from cms.models import Restaurant, Food
- r1 = Restaurant.objects.create(name='台科小炒', phone_number ='02-88888', address='台北市基隆路')
- r1.save()
- Restaurant.objects.all()

```
>>> Restaurant.objects.all()
<QuerySet [<Restaurant: 意素家>, <Restaurant: 中文小館>, <Restaurant: 台科小炒>]
```

Update

- from cms.models import Restaurant, Food
- Food.objects.all()[0].price
- Food.objects.filter(name='辣味義大利麵').update(price=200)

```
>>> Food.objects.all()[0].price  
Decimal('100')
```



```
>>> Food.objects.all()[0].price  
Decimal('200')  
>>>
```

Delete

- from cms.models import Restaurant, Food
 - Restaurant.objects.all()
-
- f = Restaurant.objects.get(name='台科小炒')
 - f.delete()

```
<QuerySet [<Restaurant: 意素家>, <Restaurant: 中文小館>, <Restaurant: 台科小炒>]>
>>>
```



```
<QuerySet [<Restaurant: 意素家>, <Restaurant: 中文小館>]>
>>>
```

CMS APP - Views

Templates

```
1 from django.shortcuts import render_to_response
2 # Create your views here.
3 from .models import Restaurant, Food
4
5 def index(request):
6     restaurants = Restaurant.objects.all()
7     return render_to_response('cms/menu.html',locals())
```

```
<h1 class="text-center">美食菜單</h1>
<div class="container-fluid">
{%
  for r in restaurants %
}
<h2>{{ r.name }}</h2>
{%
  if r.food_set.all %
}
<p>本餐廳共有{{ r.food_set.all|length }}道菜</p>
<table class="table">
  <tr>
    <th>菜名</th>
    <th>價格</th>
    <th>辣不辣</th>
    <th>註解</th>
  </tr>
  {%
    for food in r.food_set.all %
  }
  <tr>
    <td>{{ food.name }}</td>
    <td>{{ food.price }}</td>
    <td>{{ food.is_spicy }} 辣 {{ food.is_not_spicy }} 不辣 {{ food.comment }}</td>
    <td>{{ food.comment }}</td>
  </tr>
  {%
    endfor %
  }
</table>
{%
  else %
}
<p>本餐廳啥都沒賣</p>
{%
  endif %
}
{%
  endfor %
}
```

美食菜單

意素家

本餐廳共有1道菜

菜名	價格	辣不辣	註解
辣味義大利麵	200	辣	特別美食

中文小館

本餐廳啥都沒賣

Blog APP

設定APP跟時區

```
32  
33 INSTALLED_APPS = [  
34     'django.contrib.admin',  
35     'django.contrib.auth',  
36     'django.contrib.contenttypes',  
37     'django.contrib.sessions',  
38     'django.contrib.messages',  
39     'django.contrib.staticfiles',  
40     'myblog'  
41 ]  
42
```

建立Model

```
1 # -*- coding: utf-8 -*-
2 from __future__ import unicode_literals
3
4 from django.db import models
5 from django.utils import timezone
6
7
8 class Post(models.Model):
9     author = models.ForeignKey('auth.User')
10    title = models.CharField(max_length=200)
11    text = models.TextField()
12    created_date = models.DateTimeField(
13        default=timezone.now)
14    published_date = models.DateTimeField(
15        blank=True, null=True)
16
17    def publish(self):
18        self.published_date = timezone.now()
19        self.save()
20
21    def __str__(self):
22        return self.title
23
```

建立後台

Django administration

WELCOME, JERRY. [VIEW SITE / CHANGE PASSWORD / LOG OUT](#)

Home > Myblog > Posts > Add post

Add post

Author:

Title:

Text:

Created date: Date: Today |
Time: Now |

Published date: Date: Today |
Time: Now |

Save and add another Save and continue editing

Project 的 urls

```
9 Class-based views
10 |   1. Add an import: from other_app.views import Home
11 |   2. Add a URL to urlpatterns: url(r'^$', Home.as_view(), name='home')
12 Including another URLconf
13 |   1. Import the include() function: from django.conf.urls import url, include
14 |   2. Add a URL to urlpatterns: url(r'^blog/', include('blog.urls'))
15 """
16 from django.conf.urls import url, include
17 from django.contrib import admin
18
19 urlpatterns = [
20     url(r'^$', include('myblog.urls')),
21     url(r'^admin/', admin.site.urls),
22 ]
23
```

APP的urls, views

urls

```
1 from django.conf.urls import url
2 from . import views
3
4 urlpatterns = [
5     url(r'^$', views.post_list, name='post_list'),
6 ]
```

views

```
1 # -*- coding: utf-8 -*-
2 from __future__ import unicode_literals
3 from django.utils import timezone
4 from django.shortcuts import render
5 from .models import Post
6
7 # Create your views here.
8 def post_list(request):
9     posts = Post.objects.filter(published_date__lte=timezone.now()).order_by('published_date')
10    return render(request, 'blog/post_list.html', {'posts': posts})
```

建立Templates

```
1
2      {% for post in posts %}
3          <div>
4              <p>published: {{ post.published_date }}</p>
5              <h1><a href="">{{ post.title }}</a></h1>
6              <p>{{ post.text|linebreaksbr }}</p>
7          </div>
8      {% endfor %}
```

簡易Blog APP完成

published: May 6, 2017, 8:43 p.m.

將屆齡退休領18%？林萬億：一切按照制度

主導年金改革的行政院政委林萬億目前也是台大社工系教授，但有媒體消息指出，今年年滿65歲的林萬億已取得退休資格，將能享18%優惠存款利息，因而引發外界議論。林萬億對此則回應，「我是屆齡被強迫退休，也不是我自己要退的」，規定怎麼做就怎麼做。<

published: May 6, 2017, 8:43 p.m.

肚子餓了！5歲童深夜抱玩偶逛超商

新北市1名5歲的歐姓男童日前趁父母照顧弟弟睡著，深夜11時許偷溜出門，跑到住家附近的超商逗留，店員察覺有異報案，新莊所光華所員警獲報趕抵，安撫詢問後，替男童買了養樂多跟餅乾，將他帶回警局，通知父母帶回。

Album APP

設定APP與位置

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'myalbumapp'  
]
```

```
MEDIA_URL = '/media/'  
MEDIA_ROOT = os.path.join(os.path.dirname(BASE_DIR), "mideafies")
```

建立Model

```
1 # -*- coding: utf-8 -*-
2 from __future__ import unicode_literals
3 from django.db import models
4
5 class Photo(models.Model):
6     title = models.CharField(max_length=200)
7     width = models.IntegerField(default=0)
8     height = models.IntegerField(default=0)
9     image = models.ImageField(null=False, blank=False, width_field="width", height_field="height")
10    timestamp = models.DateTimeField(auto_now_add=True, auto_now=False)
11
12    def __str__(self):
13        return self.title
14
15    class Meta:
16        ordering = ["-timestamp"]
```

建立後台

Django administration

Home > Myalbumapp > Photos

Select photo to change

Action: 0 of 2 selected

<input type="checkbox"/>	TITLE	TIMESTAMP
<input type="checkbox"/>	image2	May 6, 2017, 3 p.m.
<input type="checkbox"/>	image1	May 6, 2017, 3 p.m.

2 photos

安裝Python 圖片工具

- 安裝pip3 install Pillow

```
(class2) [?] jerry > album > pip3 install Pillow
Collecting Pillow
  Downloading Pillow-4.1.1-cp36-cp36m-macosx_10_6_intel.m
  4.whl (3.5MB)
    100% |████████████████████████████████| 3.5MB 289kB/s
Collecting olefile (from Pillow)
  Using cached olefile-0.44.zip
Installing collected packages: olefile, Pillow
  Running setup.py install for olefile ... done
Successfully installed Pillow-4.1.1 olefile-0.44
```

Project 的 urls

```
from django.conf import settings
from django.conf.urls.static import static
from django.conf.urls import url, include
from django.contrib import admin

urlpatterns = [
    url(r'^admin/', admin.site.urls),
    url(r'^$', include('myalbumapp.urls')),
]

if settings.DEBUG:
    urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

APP的urls, views

urls

```
from django.conf.urls import url

from . import views

urlpatterns = [
    url(r'^$', views.photo_list, name='photo_list'),
```

views

```
from __future__ import unicode_literals

from django.shortcuts import render
from .models import Photo
# Create your views here.

def photo_list(request):
    queryset = Photo.objects.all()
    context = {
        "photos": queryset,
    }
    return render(request, "photo.html", context)
```

建立Templates



```
13 <div class="jumbotron text-center">
14 | <h1>My Photos</h1>
15 </div>
16
17 <div class="container">
18 {% for photo in photos %}
19 {% if photo.image %}
20 <h1>{{ photo.title }}</h1>
21 <img src='{{ photo.image.url }}' class="img-responsive" />
22 {% endif %}
23 {% endfor %}
24 </div>
25
26 </body>
27 </html>
```

成果

My Photos

image2



image1



dsjerry2017@gmail.com
jerry@mail.ntust.edu.tw

