Menu

Contents

[1. Cài đặt 2](#_Toc528155403)

[2. Django – Tạo project Django 2](#_Toc528155404)

[Tạo project 2](#_Toc528155405)

[Run server 2](#_Toc528155406)

[Tạo web app 2](#_Toc528155407)

[Tạo view 3](#_Toc528155408)

[3. Thiết lập cơ sở dữ liệu 5](#_Toc528155409)

[Tạo mô hình dữ liệu cho ứng dụng 5](#_Toc528155410)

[Tạo bảng từ mô hình dữ liệu 6](#_Toc528155411)

[Thao tác với các bảng 6](#_Toc528155412)

# 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'),

}

}

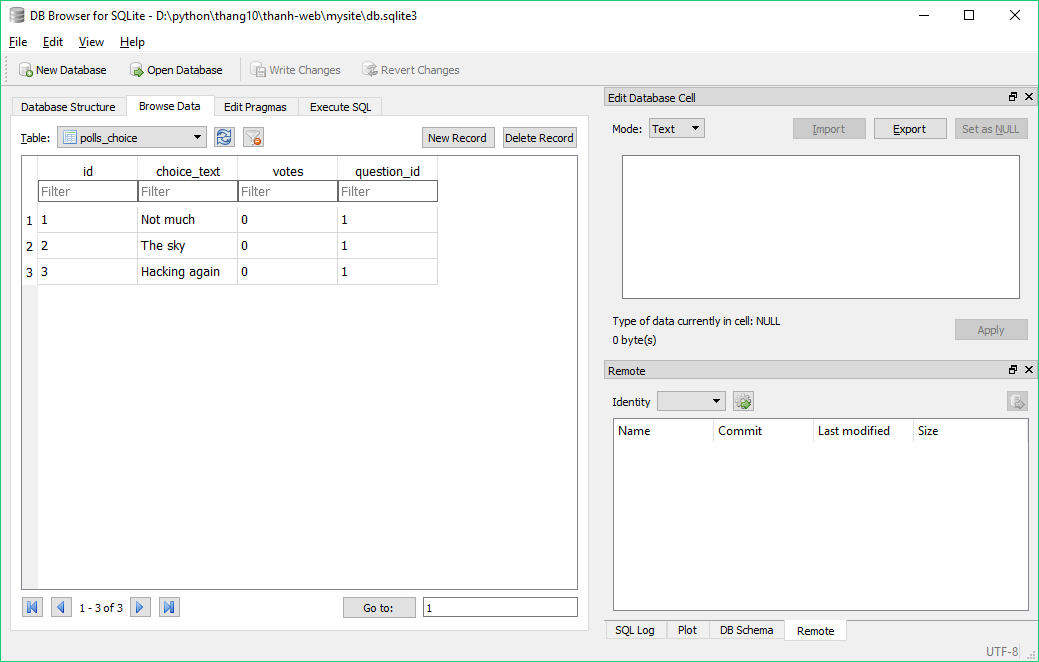
* ENGINE: tên module dành cho từng CSDL, mặc định sử dụng SQLite
  + django.db.backends.sqlite3 – cơ sở dữ liệu  SQLite
  + django.db.backends.postgresql – cơ sở dữ liệu PostgreSQL
  + django.db.backends.mysql – cơ sở dữ liệu MySQL
  + django.db.backends.oracle – cơ sở dữ liệu Oracle
* NAME: tên CSDL, 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.

Để quản lý CSDL SQLite có thể sử dụng add-on của Firefox, Chrome hoặc sử dụng chương trình như DB Browser for SQLite (sqlitebrowser.org).



## 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()

# Backend System

## Tạo user

Tạo tài khoản admin sử dụng lệnh:

python manage.py createsuperuser

Điền thông tin cần thiết như username, email, password

Username: admin

Email address: admin@example.com

Password:

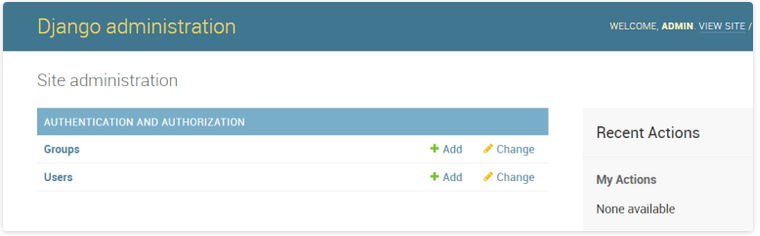
Password (again):

Superuser created successfully.

Sử dụng tài khoản vừa tạo đăng nhập vào trang quản trị với đường dẫn:

<http://127.0.0.1:8000/admin>

Trang quản trị sau khi đăng nhập thành công



Để đăng ký các bảng (hay các mô hình) với admin, sử dụng phương thức admin.site.register() trong file admin.py mà Django đã tạo cho chúng ta.

polls/admin.py

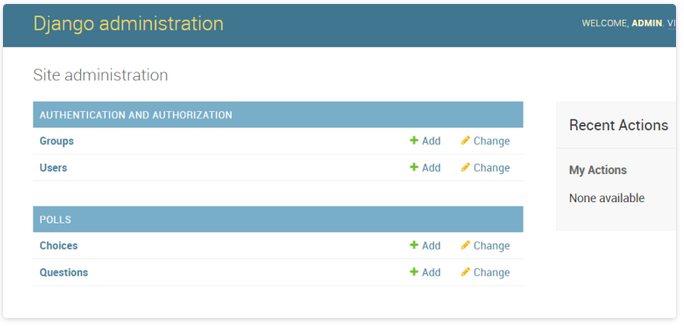
from django.contrib import admin

# Register your models here.

from .models import Question, Choice

admin.site.register(Question)

admin.site.register(Choice)



# View và Template

Trong Django thì một View là **một hàm/phương thức** làm một công việc cụ thể nào đó, một View  thường đi kèm với một Template.

Ví dụ, một ứng dụng Blog có một số View sau:

* Home – hiển thị các bài viết mới nhất.
* Entry – hiển thị bài viết nào đó.
* Archive – Lưu trữ các bài viết theo năm/tháng.
* Comment – xử lý việc đăng bình luận của độc giả.

Trong Django, một trang web được tạo ra bởi các hàm View, Django sẽ chọn View nào tùy thuộc vào URL mà chúng ta đã thiết lập.

## Tạo view

Thêm 3 view mới cho ứng dụng polls:

polls/view.py

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)

Tạo URL trỏ đến view

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'),

Ví dụ: localhost:8000/polls/34/, Django sẽ gọi đến hàm detail()

Khi gõ địa chỉ lên thanh URL của trình duyệt:

- Django sẽ đọc biến urlpatterns trong file mysite/settings.py, các đối tượng url sẽ được đọc dần dần từ trên xuống dưới cho đến khi có một đường dẫn vừa khít với URL được nhập vào.

- Khi tìm thấy localhost:8000/polls vừa khít với '^polls/', Django sẽ cắt đoạn phía sau URL ra (/polls/34/) và gửi đến file polls/urls.py để tiếp tục quá trình tìm kiếm

- Chuỗi (/polls/34/) vừa khít Regex r'^(?P<question\_id>[0-9]+)/$', => phương thức được gọi views.detail()với tham số như sau:

detail(request=<HttpRequest object>, question\_id='34')

Chuỗi Regex (?P<question\_id>[0-9]+) cho Django biết có một chuỗi con trong đoạn URL có dạng một con số (có 1 hoặc nhiều chữ số) và biến được truyền vào phương thức detail() có tên là question\_id.

## Template

Các hàm View trả về một trong 2 thứ: hoặc là trả về một đối tượng HttpReponse chứa nội dung HTML để hiển thị lên trình duyệt, hoặc là một lỗi exception 404.

Sửa lại hàm index() như sau:

polls/views.py

from django.http import HttpResponse

from .models import Question

def index(request):

    latest\_question\_list = Question.objects.order\_by('-pub\_date')[:5]

    output = ', '.join([q.question\_text for q in latest\_question\_list])

    return HttpResponse(output)

Đoạn code trên lấy ra 5 đối tượng Question được thêm vào gần đây nhất, sau đó lấy thuộc tính question\_text đưa vào biến output rồi trả về.

Khi chạy với địa chỉ localhost:8000/polls/ thì dữ liệu sẽ in ra với định dạng thô, chưa được định dạng. Ta có thể định dạng dữ liệu ngay trong view bằng các thẻ hay CSS, tuy nhiên khó quản lý và code lộn xộn.

Django cung cấp cho chúng ta hệ thống Template, chúng ta có thể viết phần giao diện ở Template và dùng nó để hiển thị dữ liệu của các View cho đẹp mắt hơn, mục đích chính của View chính là kết nối giữa Template và Model.

Đầu tiên chúng ta tạo một thư mục có tên là templates trong thư mục polls, Django sẽ tự động tìm các file template trong thư mục này. Ở đây bạn tạo thêm một thư mục khác nữa tên là tem1, trong thư mục này bạn tạo một file tên index.html.

polls/templates/temp1/index.html

{% if latest\_question\_list %}

    <ul>

       {% for question in latest\_question\_list %}

       <li>

             <a href="/polls/{{question.id}}/">

                {{question.question\_text}}

             </a>

       <li>

       {% endfor %}

    </ul>

{% else %}

   <p>No polls are available.</p>

{% endif %}

Thẻ template bắt đầu và kết thúc bằng cặp kí tự {% %} hoặc {{ }}, các câu lệnh Python nằm trong cặp dấu {% %}, còn các biến thì nằm trong cặp {{ }}.

Sửa lại đoạn code trong hàm index() để sử dụng template.

polls/views.py

#để sử dụng template

**from django.shortcuts import render**

from django.http import HttpResponse

from django.template import loader

from .models import Question

def index(request):

    latest\_question\_list = Question.objects.order\_by('-pub\_date')[:5]

#template sử dụng (từ thư mục templates)

**template = loader.get\_template('tem1/index.html')**

**context** = {

        'latest\_question\_list': latest\_question\_list,

    }

    return HttpResponse(**template.render(context, request)**)

Tham số đầu tiên là nội dung HTML trả về, tham số thứ 2 là đối tượng request được gửi đến.

Hoặc cũng có thể trả về template trực tiếp trong hàm render() như sau

polls/views.py

from django.shortcuts import render

def index(request):

    latest\_question\_list = Question.objects.order\_by('-pub\_date')[:5]

    context = {

        'latest\_question\_list': latest\_question\_list,

    }

    return **render(request, 'temp1/index.html', context)**

Với tham số thứ nhất là đối tượng request, tham số thứ 2 là đường dẫn đến file template, tham số thứ 3 là nội dung HTML trả về.

## Xử lý cho lỗi 404

polls/views.py

from django.shortcuts import render

**from django.http import HttpResponse, Http404**

from django.template import loader

from .models import Question

def detail(request, question\_id):

**try:**

        question = Question.objects.get(pk=question\_id)

**except Question.DoesNotExist:**

**raise Http404("Question does not exist")**

    return render(request, 'tem1/detail.html', {'question':question})

Phương thức get() sẽ trả về một exception DoesNotExist nếu không tìm thấy bản ghi nào trong list, tại đây chúng ta giải phóng một đối tượng Http404.

polls/templates/temp1/detail.html

<h1>{{ question.question\_text }}</h1>

<ul>

    {% for choice in question.choice\_set.all %}

        <li>{{ choice.choice\_set }}</li>

    {% endfor %}

</ul>

## URL động

Trong các file template như index.html, các đường link chúng ta tham chiếu đến trong các thẻ <a> là do chúng ta tự viết

<li>

    <a href="/tem1/{{ question.id }}/">

        {{ question.question\_text }}

    </a>

</li>

Giả sử trong file template có hàng trăm thẻ a như vậy và do vậy khi thay đổi đường dẫn giả sử tem1 thành temp/abc/tem2 thì phải thay thủ công rất mất thời gian. Cách giải quyết hiệu quả là sử dụng tham số name trong url.

Khi chúng ta tạo các đối tượng url trong file urls.py, tham số thứ 3 là name, tham số này do chúng ta tự đặt, và Django cho phép chúng ta tham chiếu đến chúng trong các file template.

<a href="{% url 'detail' question.id %}">

    {{ question.question\_text }}

</a>

Khi nào cần thay đổi URL mới thì chúng ta chỉ cần thay đổi trong file urls.py là được:

polls/urls.py

...

url(r'^details/(?P<question\_id>[0-9]+)/$', views.detail, name='detail'),

...

## Namespace cho URL

Khi dùng đến URL động thì lại phát sinh một vấn đề nữa, mặc định thì Django tự động tìm các file template bên trong thư mục template, vậy thì giả sử khi chúng ta có thêm nhiều ứng dụng khác ngoài polls, chẳng hạn như một ứng dụng blog, trong đó cũng có hàm view detail(), và hàm view này cũng sử dụng một template tên là detail.html, vậy thì khi đó Django sẽ gắn template của ứng dụng polls vào view detail() của ứng dụng blog,  như thế sẽ báo lỗi vì ứng dụng blog sẽ không có các biến giống như polls.

Để giải quyết vấn đề này, chúng ta sẽ đặt namespace cho các biến url

polls/urls.py

from django.conf.urls import url

from . import views

app\_name = "polls"

urlpatterns = [

    # ex: /polls/

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

    # ex: /polls/detils/5/

    #url(r'^(?P<question\_id>[0-9]+)/$', views.detail, name='detail'),

    url(r'^details/(?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')

]

Để đặt tên namespace cho các đối tượng url thì chúng ta chỉ cần đặt giá trị cho biến app\_name trong file urls.py là được.

polls/templates/tem1/index.html

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

    {{ question.question\_text }}

</a>

Trong file template, vd như index.html thì chúng ta chỉ cần viết đủ tên <namespace>:<tên\_biến\_url> là xong, vd url có name là detail thì viết đầy đủ thành polls:detail.