



1

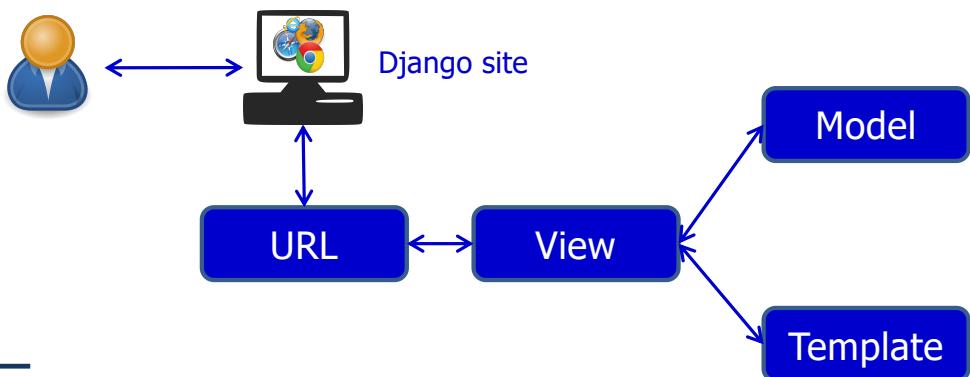
The slide shows a vertical list of topics for the tutorial, each preceded by a white circle icon. The topics are: Giới thiệu Django, Tạo project Django đầu tiên, Model và truy vấn dữ liệu, View, Admin, Truy vấn nâng cao, and Django Authentication. The slide has a similar blue-themed design with hands interacting with a digital interface in the background. The "django" logo is in the top right corner. A small number "2" is located at the bottom right corner.

Giới thiệu Django
Tạo project Django đầu tiên
Model và truy vấn dữ liệu
View
Admin
Truy vấn nâng cao
Django Authentication

2

Giới thiệu Django

- Django là một framework mạnh mẽ phát triển dễ dàng, nhanh chóng các ứng dụng Web phức tạp. Django hoạt động theo mô hình MVT (Model View Template)



Dương Hữu Thành

3

Tạo project django đầu tiên

- Cài đặt môi trường

```
pip install django
```

- Tạo project

```
django-admin startproject <project-name>
```

- Tạo app

```
django-admin startapp <app-name>
```

Dương Hữu Thành

4

4

Tạo project django đầu tiên

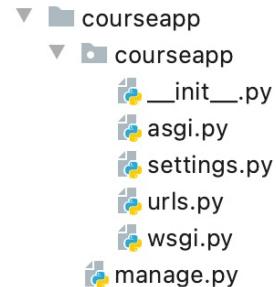
- Tạo project tên courseapp

```
django-admin startproject courseapp
```

- Chạy project

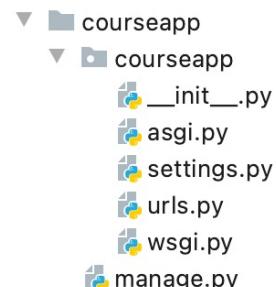
```
python manage.py runserver
```

- Truy cập <http://127.0.0.1:8000/>



Tạo project django đầu tiên

- Thư mục **courseapp/** ngoài cùng không ảnh hưởng project nên có thể đổi tên tùy thích.
- **manage.py:** một tiện ích dạng dòng lệnh cho phép tương tác project django theo nhiều cách khác nhau
- **courseapp/settings.py:** chứa thông tin cấu hình django project.





Tạo project django đầu tiên

- Một số setting quan trọng
 - **DEBUG**: thiết lập True để bật chế độ debug, nó có thể hiển thị trang web lỗi chi tiết trong quá trình phát triển. Thường khi triển khai ứng dụng production thì tắt thuộc tính này đi.
 - **INSTALLED_APPS**: danh sách các app hoạt động trong django.
 - **DATABASES**: chứa cấu hình tất cả các CSDL được sử dụng trong django.

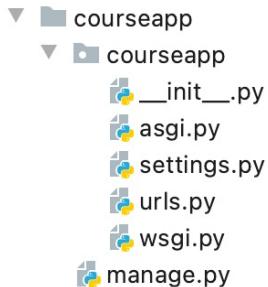


Tạo project django đầu tiên

- Một số setting quan trọng
 - **ALLOW_HOST**: danh sách các chuỗi là tên host hoặc domain được phép sử dụng dịch vụ của django site. Thuộc tính này thường rỗng khi DEBUG=True.
 - **APPEND_SLASH**: nếu là True thì các URL không kết thúc bằng slash (/). HTTP được redirect tới cùng URL như kèm slash ở cuối.
 - **AUTH_USER_MODEL**: chỉ định lớp model đại diện cho một User, mặc định là “auth.User”.

Tạo project django đầu tiên

- **courseapp/urls.py**: khai báo các URLs cho django project.
- **courseapp/asgi.py**: dùng cho các web server tương thích ASGI (Asynchronous Server Gateway).
- **courseapp/wsgi.py**: dùng cho các web server tương thích WSGI (Web Server Gateway Interface).



Dương Hữu Thành

9

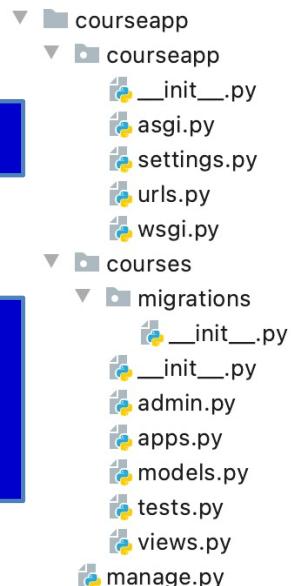
- Tạo app trong project

```
django-admin startapp courses
```

- Viết courses/views.py

```
from django.http import HttpResponse

def index(request):
    return HttpResponse("e-Course App")
```



Dương Hữu Thành

10

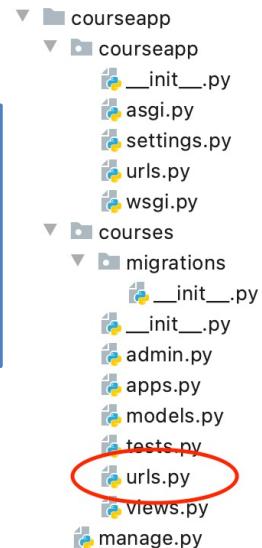
10

Tạo project django đầu tiên

- Tạo courses/urls.py

```
from django.urls import path
from . import views

urlpatterns = [
    path('', views.index, name="index")
]
```



- Chính sửa courseapp/urls.py

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('', include('courses.urls')),
    path('admin/', admin.site.urls),
]
```

- **include** dùng tham chiếu URLconfs khác.



Kết nối cơ sở dữ liệu

- Biến **DATABASES** trong settings.py dùng cấu hình kết nối CSDL, mặc định database được cấu hình sử dụng **SQLite** có sẵn trong.

```
DATABASES = {  
    'default': {  
        'ENGINE': 'django.db.backends.sqlite3',  
        'NAME': BASE_DIR / 'db.sqlite3',  
    }  
}
```

- Chú ý nếu sử dụng engine không phải SQLite thì phải **đảm bảo** CSDL chỉ định đã **có sẵn**.



Kết nối cơ sở dữ liệu

- ENGINE: chỉ định engine hệ quản trị cơ sở dữ liệu sẽ sử dụng
 - django.db.backends.sqlite3
 - django.db.backends.mysql
 - django.db.backends.postgresql
- NAME: tên cơ sở dữ liệu
- USER: username của cơ sở dữ liệu.
- PASSWORD: password của cơ sở dữ liệu.
- HOST: host chứa cơ sở dữ liệu.

Kết nối cơ sở dữ liệu

- Ví dụ cấu hình kết nối đến cơ sở dữ liệu MySQL có tên là coursedb, username là root, password là 12345678.

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'coursedb',
        'USER': 'root',
        'PASSWORD': '12345678',
        'HOST': '' # mặc định localhost
    }
}
```

- Mở terminal cài đặt mysql driver

```
pip install mysqlclient
```

- Tham khảo cách cài đặt trên các nền tảng khác
 - <https://pypi.org/project/mysqlclient/>



Model

- Một model là nơi duy nhất thực sự tương tác với dữ liệu. Mỗi model là một class kế thừa – `django.db.models.Model`
- Model chứa các trường (field) và hành vi (behavior) dữ liệu được lưu trữ.
- Mỗi trường trong model đại diện cho một trường của bảng trong CSDL.

Dương Hữu Thành

17



Model

- Mỗi trường của model là thể hiện lớp Field, đây là lớp trừu tượng có nhiều lớp con cho các kiểu dữ liệu tương ứng như:
 - CharField
 - TextField
 - DateTimeField
 - BooleanField
 - SlugField
 - JSONField

Dương Hữu Thành

18



Model

- Tạo các lớp model trong courses/models.py

```
from django.db import models

class Category(models.Model):
    name = models.CharField(max_length=100,
                           unique=True)

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



Model

- Thiết lập khoá ngoại

```
class Course(models.Model):
    subject = models.CharField(max_length=100, unique=True)
    description = models.CharField(max_length=255)
    created_date = models.DateTimeField(auto_now_add=True)
    updated_date = models.DateTimeField(auto_now=True)
    category = models.ForeignKey(Category,
                                 on_delete=models.CASCADE)
    active = models.BooleanField(default=True)

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

Model

- Ta cần cho django biết về sự tồn tại của courses app thông qua biến **INSTALLED_APP** trong tập tin cấu hình settings.py.

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

Dương Hữu Thành

21

21

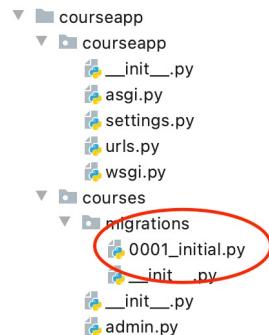
Model

- Lệnh **makemigrations** cho django biết **có sự thay đổi** trong models.py và ta muốn lưu các thay đổi số trong **migration**.

```
python manage.py makemigrations courses
```

Migrations for 'courses':

courses/migrations/0001_initial.py
- Create model Category
- Create model Course



Dương Hữu Thành

22

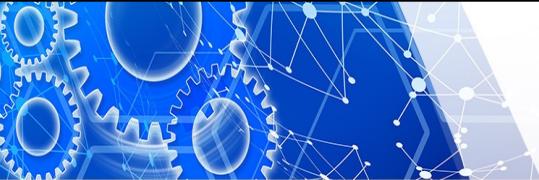
22



Model

- Ta cũng có thể sử dụng lệnh **sqlmigrate** để xem SQL sẽ được tạo ra từ một migration để thực thi trong cơ sở dữ liệu.

```
python manage.py sqlmigrate courses 0001
```



Model

- Lệnh **migrate** thực thi migration để áp dụng những thay đổi trong models xuống lược đồ cơ sở dữ liệu.

```
python manage.py migrate
```

- Một migration được thực thi sẽ được lưu trong tin trong bảng **django_migrations** và nó sẽ không chạy lại lần sau.
- Lệnh migrate chỉ chạy cho các app định nghĩa trong biến INSTALLED_APP.

Model

Dương Hữu Thành

25

coursesdb
Tables
auth_group
auth_group_permissions
auth_permission
auth_user
auth_user_groups
auth_user_user_permissions
courses_category
courses_course
django_admin_log
django_content_type
django_migrations
django_session

courses_category
Columns
id
name
Indexes
Foreign Keys
Triggers
courses_course
Columns
id
subject
description
created_date
updated_date
active
category_id
Indexes
Foreign Keys
Triggers

Dương Hữu Thành

26

django_migrations

1 • `SELECT * FROM coursedb.django_migrations;`

id	app	name	applied
4	admin	0002_logentry_remo...	2021-05-28 15:58:09.513633
5	admin	0003_logentry_add_...	2021-05-28 15:58:09.525912
6	contenttypes	0002_remove conte...	2021-05-28 15:58:09.602613
7	auth	0002_alter_permissi...	2021-05-28 15:58:09.643749
8	auth	0003_alter_user_em...	2021-05-28 15:58:09.672321
9	auth	0004_alter_user_us...	2021-05-28 15:58:09.684635
10	auth	0005_alter_user_las...	2021-05-28 15:58:09.728184
11	auth	0006_require conte...	2021-05-28 15:58:09.730515
12	auth	0007_alter_validator...	2021-05-28 15:58:09.741286
13	auth	0008_alter_user_us...	2021-05-28 15:58:09.793604
14	auth	0009_alter_user_las...	2021-05-28 15:58:09.835678
15	auth	0010_alter_group_n...	2021-05-28 15:58:09.853374
16	auth	0011_update_proxy...	2021-05-28 15:58:09.865912
17	auth	0012_alter_user_firs...	2021-05-28 15:58:09.915457
18	courses	0001_initial	2021-05-28 15:58:09.979292
19	sessions	0001_initial	2021-05-28 15:58:10.007904



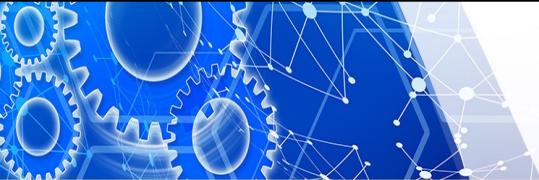
Model

- Làm việc trên shell: `python manage.py shell`

```
>>> from courses.models import *
>>> c = Category(name="Công nghệ Thông tin")
>>> c.save()
>>> co = Course(subject="Lập trình Python", category=c)
>>> co.save()
>>> co.subject = "Các công nghệ lập trình hiện đại"
>>> co.save()
>>> Course.objects.all()
<QuerySet [<Course: Các công nghệ lập trình hiện đại>]>
>>> Course.objects.filter(subject__contains="hiện đại")
<QuerySet [<Course: Các công nghệ lập trình hiện đại>]>
>>> Course.objects.get(pk=1)
<Course: Các công nghệ lập trình hiện đại>
```

Dương Hữu Thành

27



Model

- Khi truy vấn muốn xem câu truy vấn SQL được tạo ra sử dụng đối tượng query như sau:

```
>>> q = Course.objects.filter(subject__icontains='các')
>>> print(q.query)
SELECT `courses_course`.`id`,
`courses_course`.`subject`,
`courses_course`.`description`,
`courses_course`.`created_date`,
`courses_course`.`updated_date`,
`courses_course`.`category_id`,
`courses_course`.`active` FROM `courses_course` WHERE
`courses_course`.`subject` LIKE %các%
```

Dương Hữu Thành

28

28



Một số thuộc tính lớp Field

- **db_column**: chỉ định tên cột trong cơ sở dữ liệu, mặc định sẽ sử dụng tên trường.
- **primary_key**: chỉ định trường khoá chính, nếu không có trường nào primary_key thì django sẽ tự tạo một khoá chính auto increase.
- **null** (mặc định false) cho phép trường null.
- **blank** (mặc định false) cho phép trường rỗng.
- **default**: chỉ giá trị mặc định.
- **unique**: chỉ định giá trị duy nhất trong bảng.



Một số thuộc tính lớp Field

- **db_index**: nếu True thì trường sẽ được index trong cơ sở dữ liệu.
- **editable**: nếu là False sẽ không hiển thị trong admin, mặc định là True.
- **help_text**: văn bản hướng dẫn hiển thị trong form widget.
- **error_messages**: cho phép ghi đè một số thông điệp lỗi, giá trị nó là dict có key khớp với message muốn ghi đè như null, blank, invalid, unique, invalid_choice, unique_for_date.



Một số thuộc tính lớp Field

- **choices**: dùng cho các trường có nhiều lựa chọn, mặc định trong form widget sử dụng thẻ select, giá trị cho thuộc tính này là dãy các tuple, mỗi tuple:
 - Giá trị đầu của tuple là giá trị thật sự thiết lập cho model.
 - Giá trị thứ hai của tuple là tên sử dụng thân thiện cho người dùng.



Một số Field thông dụng

- **CharField**: lưu chuỗi nhỏ, để lưu nhiều văn bản lớn thì dùng **TextField** thay thế.
 - max_length: chiều dài tối đa (bắt buộc)
- Một số field đặc biệt của CharField: EmailField, URLField, SlugField.
- **BooleanField**: lưu giá trị luận lý true/false, mặc định giá trị là None.



Một số Field thông dụng

- **IntegerField**: lưu số nguyên, giá trị -2147483648 đến 2147483647.
- **FloatField**: lưu số chấm động (floating-point), trong python là thể hiện của float.
- **DecimalField**: lưu số chấm tĩnh (fixed-precision decimal number), trong python là thể hiện của Decimal.
 - max_digits: số chữ số tối đa.
 - decimal_places: số chữ số thập phân, giá trị phải nhỏ hơn hoặc bằng max_digits.



Một số Field thông dụng

- **DateField**: dữ liệu date (trong python là thể hiện của datetime.date)
 - auto_now: trường tự động thiết lập là ngày hiện tại khi đối tượng được lưu (khi gọi phương thức Model.save()).
 - auto_now_add: trường tự động thiết lập là ngày hiện tại khi đối tượng lần đầu được tạo.
- **DateTimeField**: lưu trữ data và time (trong python là thể hiện của datetime.datetime).



Một số Field thông dụng

- **JSONField**: lưu trữ dữ liệu dạng json.
- **SlugField**: slug là một nhãn ngắn gọn chỉ chứa các ký tự, ký số, dấu gạch chân (underscores), gạch giữa (hyphens). Slug thường sử dụng tạo các URL thân thiện SEO, mặc định db_index là True và max_length=50.
- **UUIDField**: trường lưu trữ các chuỗi định dạng duy nhất.



Một số Field thông dụng

- **FileField**: trường để upload tập tin
 - upload_to: thư mục chứa tập tin upload.
 - max_length: chiều dài tối đa đường dẫn.
- **ImageField**
 - Kế thừa FileField dùng upload hình ảnh.
 - Được tạo trong CSDL là cột kiểu varchar, chiều dài tối đa 100.
 - Để sử dụng ImageField, yêu cầu cài pillow ([pip install Pillow](#)).



Meta options của model

- **abstract**: thiết lập lớp model là trừu tượng.
- **app_label**: tên app chứa model, nếu models được khai báo ngoài app thì thuộc tính này bắt buộc phải khai báo.
- **db_table**: tên bảng CSDL sẽ được tạo cho models, mặc định {app_name}_{model_name} với các ký tự thường (lowercase).

Dương Hữu Thành

37

37



Meta options của model

- **ordering**: chỉ định các trường dùng sắp xếp khi truy vấn dữ liệu. Mặc định là sắp xếp tăng, nếu thêm dấu “-” trước tên trường là sắp xếp giảm.
- **unique_together**: thiết lập ràng buộc giá trị duy nhất cho nhiều trường kết hợp.
- **constraints**: thiết lập một số ràng buộc trong CSDL.

Dương Hữu Thành

38

38



Meta options của model

- Ví dụ các model đều cần thông tin created_date, updated_date, active và một trường lưu ảnh đại diện. Ta có thể tách thành model trừu tượng chung và model Course kế thừa lại
- Mặc định các model kế thừa lại model này sẽ có dữ liệu sắp giảm theo id khi truy vấn.
- Giả sử quy định thêm trong cùng một danh mục không được phép có hai khoá học trùng tên.

Dương Hữu Thành

39

39



Meta options của model

```
class ModelBase(models.Model):  
    created_date = models.DateTimeField(auto_now_add=True)  
    updated_date = models.DateTimeField(auto_now=True)  
    active = models.BooleanField(default=True)  
    image = models.ImageField(upload_to='courses/%Y/%m/' ,  
                             null=True, blank=True)  
  
    class Meta:  
        abstract = True  
        ordering = ['-id'] # sắp giảm theo id  
  
class Course(ModelBase):  
    class Meta:  
        unique_together = ('subject', 'category')
```

Dương Hữu Thành

40

40

Meta options của model

- Chú ý trường ImageField khi thực thi django sẽ kết hợp MEDIA_ROOT trong settings.py để xác định nơi upload ảnh, đường dẫn có dạng **MEDIA_ROOT/<giá trị thuộc tính upload_to>**
- Cấu hình biến MEDIA_ROOT trong settings.py

```
MEDIA_ROOT = '%s/courses/static/' % BASE_DIR
```

Meta options của model

- Truy cập lại trang admin quản lý khoá học, tiến hành sửa hoặc tạo khoá học mới sẽ có thêm trường cho phép upload hình ảnh.

Change course

Các công nghệ lập trình hiện đại

Image: No file chosen

Active

Subject: Các công nghệ lập trình hiện đại

Description: Các công nghệ lập trình hiện đại

Category: Công nghệ Thông tin



Quan hệ ManyToOne

- Sử dụng lớp `models.ForeignKey` để thiết lập quan hệ ManyToOne. Lớp này yêu cầu 2 đối số:
 - model được tham chiếu tới
 - giá trị đối số cho `on_delete`
- Các trường `ForeignKey` sẽ **tự động được tạo index** (`db_index=True`).
- Khi thực hiện migrate để tạo CSDL thì django sẽ **tự thêm đuôi _id** sau tên trường `ForeignKey`. Ví dụ `category`, dưới CSDL sẽ là `category_id`.

Dương Hữu Thành

43

43



Quan hệ ManyToOne

- Giá trị `on_delete` cho biết cách xử lý nếu đối tượng được tham chiếu bởi `ForeignKey` bị xoá
 - **SET_NULL**: thiết lập null cho trường `ForeignKey`, khi đó thuộc tính `null=True` trong `ForeignKey`).
 - **SET_DEFAULT**: thiết lập giá trị mặc định, sử dụng thêm thuộc tính `default` trong `ForeignKey`.
 - **CASCADE**: các đối tượng có chứa khai báo `ForeignKey` sẽ bị xoá theo.

Dương Hữu Thành

44

44



Quan hệ ManyToOne

- **PROTECT**: ngăn không cho đối tượng được tham chiếu bị xoá, bằng cách ném ngoại lệ ProtectedError.
- **RESTRICT**: ngăn không cho đối tượng được tham chiếu bị xoá, bằng cách ném ngoại lệ RestrictedError. Nó khác PROTECT, đối tượng được tham chiếu cũng có thể được xoá, nếu nó chứa ForeignKey(on_delete=CASCADE) tới một model khác.
- **SET()**, **DO NOTHING**

Dương Hữu Thành

45

45



Quan hệ ManyToOne

- Ví dụ tạo thêm một model Lesson. Trong đó một bài học (Lesson) phải thuộc một khoá học (Course), trong khoá học có thể có nhiều bài học. Giả sử quy định:
 - Trong cùng một khoá học thì tựa đề các bài học là duy nhất.
 - Khi các khoá học bị xoá thì các bài học của nó cũng sẽ bị xoá theo.

Dương Hữu Thành

46

46

Quan hệ ManyToOne

```
class Lesson(ModelBase):
    class Meta:
        unique_together = ('subject', 'course')

    subject = models.CharField(max_length=255)
    content = models.TextField()
    course = models.ForeignKey(
        Course, on_delete=models.CASCADE
        related_name='lessons',
        related_query_name='my_lesson'
    )

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

Dương Hữu Thành

47



Quan hệ ManyToOne

- Thực thi makemigrations courses

Migrations for 'courses':

courses/migrations/0002_lesson.py
- Create model Lesson

- Thực thi migrate

Operations to perform:

Apply all migrations: admin, auth, contenttypes, courses, sessions

Running migrations:

Applying courses.0002_lesson... OK

- Tạo một dòng dữ liệu mẫu cho lesson

```
>>> c = Course.objects.get(pk=1)
>>> Lesson.objects.create(subject="Giới thiệu",
content="Giới thiệu các công nghệ lập trình hiện đại",
course=c)
<Lesson: Giới thiệu>
```



48



Quan hệ ManyToOne

- Truy vấn trong quan hệ khoá ngoại

```
>>> courses =  
Course.objects.filter(category__name__icontains='tin')  
>>> print(courses.query)  
SELECT `courses_course`.`id`,  
`courses_course`.`created_date`,  
`courses_course`.`updated_date`,  
`courses_course`.`active`, `courses_course`.`subject`,  
`courses_course`.`description`,  
`courses_course`.`category_id` FROM `courses_course`  
INNER JOIN `courses_category` ON  
(`courses_course`.`category_id` =  
`courses_category`.`id`) WHERE  
`courses_category`.`name` LIKE %tin%
```

Dương Hữu Thành

49



Quan hệ ManyToOne

- **related_name**: chỉ định tên được sử dụng bởi đối tượng được tham chiếu khoá ngoại. Nếu ta không muốn django tạo quan hệ truy vấn ngược (backward) trong ForeignKey thì thiết lập giá trị thuộc tính này là "+" hoặc kết thúc là "+".
- **related_query_name**: tên dùng để lọc dữ liệu từ đối tượng được tham chiếu khoá ngoại (mặc định là giá trị từ related_name)

Dương Hữu Thành

50



Quan hệ ManyToOne

- Course không khai báo related_name trong ForeignKey tới Category.
- Từ Category truy vấn danh sách course của nó.

```
>>> c = Category.objects.get(pk=1)
>>> c.course_set.all() # course_set ngầm định được thêm
<QuerySet [<Course: Các công nghệ lập trình hiện đại>]>
```



Quan hệ ManyToOne

- Lesson có khai báo related_name trong ForeignKey tới Course.
- Từ Course truy vấn danh sách lesson của nó.

```
>>> c = Course.objects.get(pk=1)
>>> c.lessons.all() # lessons khai báo bởi related_name
<QuerySet [<Lesson: Giới thiệu>]>
```



Quan hệ ManyToOne

- Ví dụ sử dụng related_query_name
- Không khai báo related_query_name

```
>>> Category.objects.filter(course__active=True)
<QuerySet []>
```

- Có khai báo related_query_name

```
>>> Course.objects.filter(my_lesson__active=True)
<QuerySet []>
```



Quan hệ ManyToMany

- Sử dụng lớp ManyToManyField bắt buộc một đối số là model có liên quan trong quan hệ.
- Các thuộc tính quan trọng
 - **related_name**: tương tự ForeignKey
 - **related_query_name**: tương tự ForeignKey
 - **through**: chỉ định bảng model trung gian cho quan hệ ManyToMany.
 - **symmetrical**: sử dụng khi định nghĩa quan hệ ManyToMany đến chính nó.



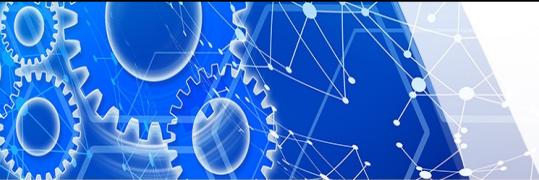
Quan hệ ManyToMany

- Django sẽ **tự động tạo bảng trung gian** của quan hệ ManyToMany dưới CSDL.
- Tuy nhiên nếu ta muốn chỉ định bảng trung gian thì sử dụng **through** chỉ định model trung gian đại bảng trung gian.

Dương Hữu Thành

55

55



Quan hệ ManyToMany

- Thêm model Tag

```
class Tag(models.Model):  
    name = models.CharField(max_length=100,  
                           unique=True)  
  
    def __str__(self):  
        return self.name
```

- Tại model Lesson thiết lập quan hệ ManyToMany

```
class Lesson(ModelBase):  
    ...  
    tags = models.ManyToManyField('Tag', blank=True,  
                                related_name='lessons')
```

Dương Hữu Thành

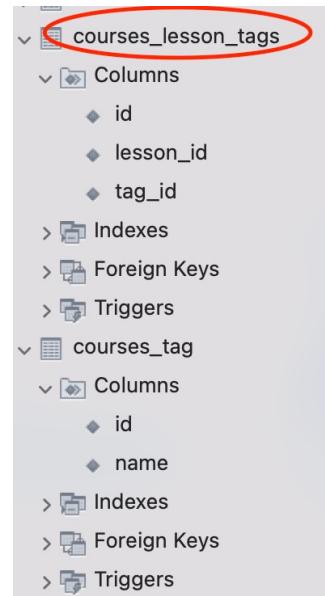
56

56



Quan hệ ManyToMany

- Tạo và thực thi migration để áp dụng vào CSDL.
- Một bảng trung gian tự động được tạo courses_lesson_tags.



57

Dương Hữu Thành



Quan hệ ManyToMany

- Thêm Tag vào Lesson

```
>>> t = Tag.objects.create(name="nhập môn")
>>> l = Lesson.objects.get(pk=1)
>>> l.tags.add(t)
>>> l.save()
```

- Xem danh sách Lesson của Tag và ngược lại.

```
>>> l.tags.all()
<QuerySet []>
>>> t.lessons.all()
<QuerySet []>
```

Dương Hữu Thành

58

58



Quan hệ giữa các model

- OneToOneField bắt buộc có đối số là tham chiếu đến model liên quan.

```
user_ptr = models.OneToOneField(User,  
on_delete=models.CASCADE, primary_key=True)
```

- Mỗi quan hệ này tương đương ForeignKey với unique=True.



Kế thừa model

- Kế thừa model về căn bản giống như kế thừa các lớp bình thường trong python.
- Có 3 kiểu kế thừa model trong django
 - Abstract base class
 - Multi-table inheritance
 - Proxy model



Abstract base class

- Abstract base class : model cha (thuộc tính abstract=True trong Meta) đơn giản chỉ dùng chứa thông tin chung cho các model con, model này sẽ **không được** tạo bảng dưới CSDL.

```
class ModelBase(models.Model):  
    class Meta:  
        abstract = True  
  
        active = models.BooleanField(default=True)  
  
    class Course(ModelBase):  
        subject = models.CharField(max_length=100)
```

Dương Hữu Thành

61

61



Multi-table inheritance

- Các model bình thường kế thừa, **mỗi model sẽ tạo một bảng riêng** dưới CSDL.
- Ví dụ có hai bảng tương ứng hai model User và MyUser được tạo ra, trong đó MyUser có đầy đủ thuộc tính User và có thêm thuộc tính avatar.

```
from django.contrib.auth.models import User  
  
class MyUser(User):  
    avatar = models.ImageField(upload_to='upload/')
```

Dương Hữu Thành

62

62



Multi-table inheritance

- Django tự động tạo ra quan hệ OneToOne giữa model con và mỗi model cha, chẳng hạn model MyUser sẽ tự động có trường như sau:

```
user_ptr = models.OneToOneField(User,
on_delete=models.CASCADE, parent_link=True,
primary_key=True)
```

- Nếu ta muốn thay đổi tên các thuộc tính này thì tạo trường OneToOneField và thiết lập thuộc tính parent_link=True để chỉ định trường đó dùng liên kết ngược lên model cha.



Proxy Model

- Những model con chỉ muốn thay đổi một số hành vi của model cha, mà không có nhu cầu thêm thuộc tính mới, thì ta có thể tạo kẽ thừa kiểu proxy model.
- Ta có thể thao tác thêm, xoá, sửa trên thể hiện của proxy model và sẽ được lưu giống như đang làm việc trên model gốc.



Proxy Model

- Ví dụ model MyGuest làm việc cùng bảng dữ liệu với model Guest.

```
class Guest(models.Model):  
    first_name = models.CharField(max_length=30)  
    last_name = models.CharField(max_length=30)  
  
class MyGuest(Guest):  
    class Meta:  
        proxy = True  
  
    def do_something(self):  
        pass
```



Truy vấn dữ liệu

- **create()**: tạo đối tượng và lưu vào CSDL.
- **update()**: cập nhật đối tượng vào CSDL.
- **delete()**: xoá đối tượng khỏi CSDL.
- **save()**: lưu đối tượng vào CSDL.



Truy vấn dữ liệu

- **get_or_create(defaults, **kwargs)**: tìm kiếm một đối tượng theo kwargs, nếu không tìm thấy sẽ tạo đối tượng thông tin tương ứng. Phương thức trả về tuple (object, created), trong đó object là đối tượng, created cho biết nó có phải tạo mới hay không.
- **updated_or_create(defaults, **kwargs)**: cập nhật đối tượng thỏa điều kiện kwargs.



Truy vấn dữ liệu

- **count()**: số đối tượng trong QuerySet.
- **latest()**: trả về đối tượng cuối trong QuerySet dựa trên trường chỉ định.
- **earliest()**: ngược lại với latest()
- **first()**: trả về đối tượng đầu tiên trong QuerySet
- **last()**: trả về đối tượng cuối trong QuerySet
- **exists()**: kiểm tra QuerySet có tồn tại kết quả nào không.
- **aggregate()**: thống kê cho QuerySet (sum, max)

Truy vấn dữ liệu

- Ví dụ

```
>>> courses = Course.objects
>>> courses.count()
8
>>> courses.first()
<Course: Cơ sở lập trình>
>>> courses.earliest('created_date')
<Course: Các công nghệ lập trình hiện đại>
>>> courses.filter(active=False).exists()
False
>>> from django.db.models import Count
>>> courses.aggregate(Count('id'))
{'id__count': 8}
```

Dương Hữu Thành

69

Truy vấn dữ liệu

- Để truy vấn CSDL, ta tạo QuerySet thông qua Manager trên lớp model. Mỗi model có ít nhất một Manager gọi là objects. Một QuerySet đại diện cho nhiều đối tượng từ CSDL.
- QuerySet là **lazy**, tức là khi tạo QuerySet nó chưa thực sự truy vấn xuống CSDL, cho đến khi có một lệnh nào đó yêu cầu thực hiện (**evaluated**).

Dương Hữu Thành

70



Truy vấn dữ liệu

- QuerySet sẽ được thực thi khi:
 - iteration: lặp qua queryset.
 - slicing: lấy một phần của queryset.
 - picking/caching: lưu lại queryset.
 - repr(): dùng xem nhanh kết quả QuerySet.
 - len(): lấy số lượng phần tử queryset.
 - list(): chuyển QuerySet thành list.
 - bool(): dùng QuerySet để kiểm tra điều kiện nào đó.

Dương Hữu Thành

71

71



Truy vấn dữ liệu

- Lấy **tất cả** các đối tượng

```
Course.objects.all()
```

- Sử dụng **filter** để lọc dữ liệu

```
# Lấy các course có năm tạo là 2021  
Course.objects.filter(created_date__year=2021)  
# Lấy các course có tên chủ đề bắt đầu bằng "Các"  
Course.objects.filter(subject__startswith="Các")
```

- Sử dụng **exclude** để loại bỏ dữ liệu

```
# Lấy các course trừ course có năm tạo <= 2020  
Course.objects.exclude(created_date__year__lte=2020)
```

Dương Hữu Thành

72

72



Truy vấn dữ liệu

- Mặc định các điều kiện trong filter hoặc exclude sẽ liên kết bằng phép AND. Để thực hiện các truy vấn phức tạp hơn sử dụng `Q()`. Các Q có thể kết hợp với nhau bằng phép & hoặc |.

```
Course.objects.filter(Q(created_date__year=2020) | \  
                      ~Q(subject__icontains='lập trình'))
```



Truy vấn dữ liệu

- Các phương thức filter và exclude có thể thực hiện **nối tiếp nhau thành chuỗi** (chain) truy vấn.
- Ví dụ lấy các khoá học có subject chứa từ "hiện đại", active là True, nhưng trừ những khoá có năm tạo nhỏ hơn hoặc bằng 2020.

```
Course.objects.filter(subject__contains="hiện đại",  
                      active=True) \  
                      .exclude(created_date__lte=2020)
```



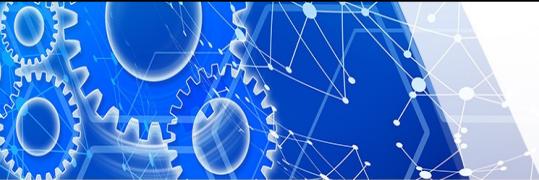
Truy vấn dữ liệu

- Một số field lookups chỉ định điều kiện
 - exact và iexact
 - contains và icontains
 - in
 - gt, gte, lt, lte, range
 - startswith, istartswith, endswith, iendswith
 - regex, iregex
 - date, year, month, day, hour, minute, second

Dương Hữu Thành

75

75



Truy vấn dữ liệu

```
>>> Course.objects.filter(subject__contains='Hiện đại')
<QuerySet []>
>>> Course.objects.filter(subject__icontains='Hiện đại')
<QuerySet [<Course: Các công nghệ lập trình hiện đại>]>
>>> Course.objects.filter(pk__in=[1, 2, 5])
<QuerySet [<Course: Công nghệ phần mềm>, <Course: Lập trình Java>, <Course: Các công nghệ lập trình hiện đại>]>
>>> Course.objects.filter(subject__endswith='phần mềm',
    updated_date__month=6)
<QuerySet [<Course: Công nghệ phần mềm>, <Course: Kiểm thử phần mềm>]>
>>> Course.objects.filter(created_date__month__range=[6, 9],
    subject__iregex='(giải thuật|thuật giải)')
<QuerySet [<Course: Cấu trúc dữ liệu và giải thuật>]>
```

Dương Hữu Thành

76

76



Truy vấn dữ liệu

- **order_by()**: thực hiện sắp xếp, mặc định QuerySet sắp xếp dựa trên thuộc tính ordering trong Meta của model, ta cũng có thể ghi đè bằng cách sử dụng order_by().
- Ví dụ sắp xếp các khoá học giảm dần theo mã danh mục, nếu cùng mã danh mục thì tăng dần theo id khoá học

```
Course.objects.order_by('category_id', '-id')
```



Admin

- Tạo user vai trò admin

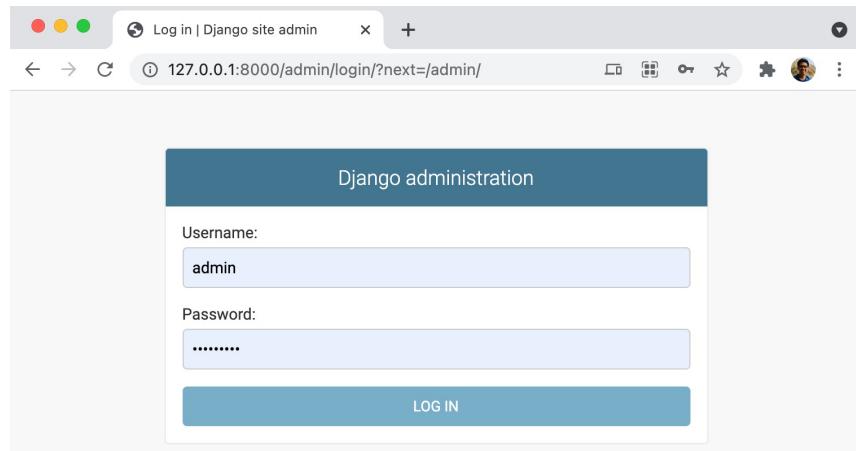
```
python manage.py createsuperuser
```

- Sau lệnh này nhập các thông tin: username, email, password. Khi đó một user sẽ được tạo trong bảng auth_user.

```
Username (leave blank to use 'duonghuuthanh'): admin
Email address: dhthanhqa@gmail.com
Password: *****
Password (again): *****
Superuser created successfully.
```

Admin

- Truy cập <http://127.0.0.1:8000/admin> và đăng nhập với thông tin user vừa tạo.



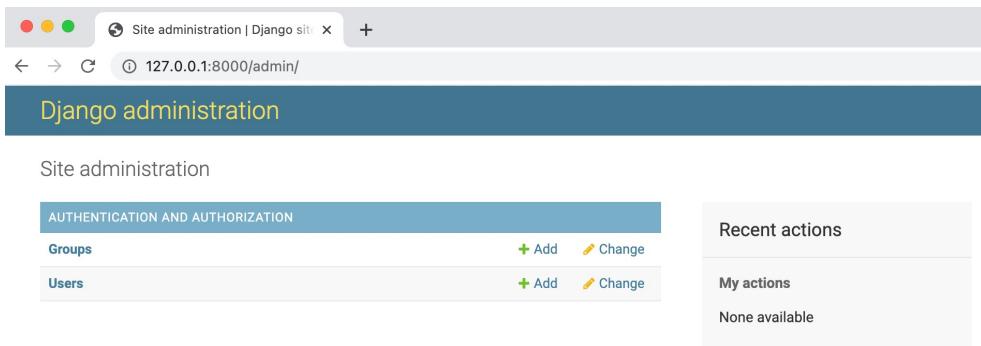
79

Dương Hữu Thành

79

Admin

- Trang admin mặc định của django



80

Dương Hữu Thành

80

Admin

- Thêm hai model là Category và Course vào admin, chỉnh sửa courses/admin.py như sau:

```
from django.contrib import admin
from .models import Category, Course

admin.site.register(Category)
admin.site.register(Course)
```

Dương Hữu Thành

81

81

The screenshot shows the Django Admin interface. The top navigation bar says "Django administration" and "Admin". The left sidebar has sections for AUTHENTICATION AND AUTHORIZATION (Groups, Users) and COURSES (Categorys, Courses). The "Courses" section is highlighted with a red oval. The main content area shows a "Recent actions" sidebar with "My actions: None available". Below it is a "Select course to change" form with a "Select course to change" title, an "Action:" dropdown, a "Go" button, and a message "0 of 1 selected". There are two checkboxes: "COURSE" and "Cács công nghệ lập trình hiện đại". A note at the bottom says "1 course".

Dương Hữu Thành

82

82

ModelAdmin

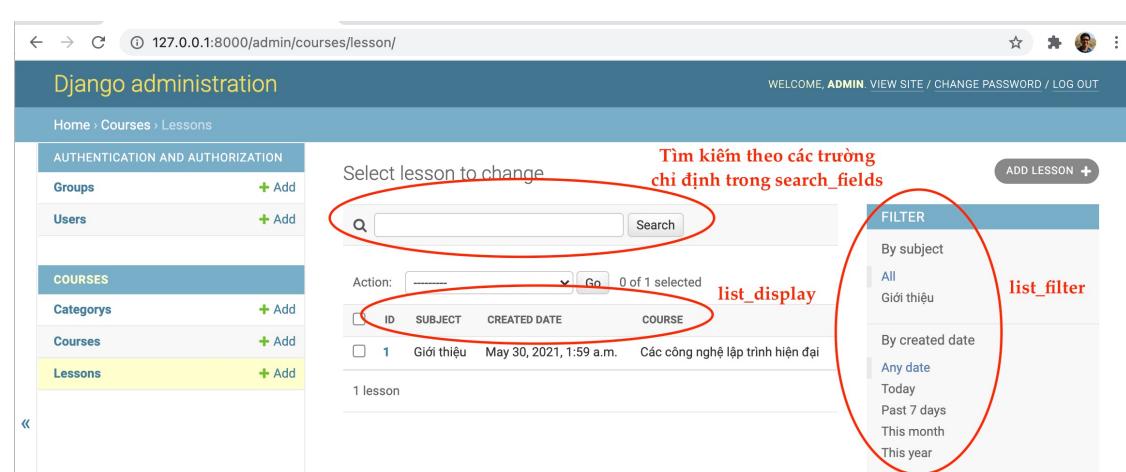
- Bổ sung courses/admin.py trang quản lý Lesson

```
class LessonAdmin(admin.ModelAdmin):  
    list_display = ['id', 'subject',  
                   'created_date', 'course']  
    list_filter = ['subject', 'created_date']  
    search_fields = ['subject', 'course__subject']  
  
admin.site.register(Lesson, LessonAdmin)
```

Dương Hữu Thành

83

ModelAdmin



The screenshot shows the Django admin interface for the 'Lessons' model. The URL is 127.0.0.1:8000/admin/courses/lesson/. The sidebar on the left has sections for AUTHENTICATION AND AUTHORIZATION, COURSES, and LESSONS. The LESSONS section is highlighted. The main content area shows a table with one row of data: ID 1, SUBJECT Giới thiệu, CREATED DATE May 30, 2021, 1:59 a.m., COURSE Các công nghệ lập trình hiện đại. At the top right, there are search and filter tools. Red circles highlight the search bar and the filter dropdown. Red annotations explain the search bar as 'Tim kiếm theo các trường chỉ định trong search_fields' and the filter dropdown as 'list_filter'.

Dương Hữu Thành

84

84

ModelAdmin

- Chỉnh sửa courses/admin.py cho phép hiển thị ảnh đã upload trong trang admin.

```
from django.utils.html import mark_safe
class CourseAdmin(admin.ModelAdmin):
    list_display = ['id', 'subject', 'description']
    readonly_fields = ['avatar']

    def avatar(self, obj):
        if obj:
            return mark_safe(
                '\'\
                .format(url=obj.image.name)
            )

admin.site.register(Course, CourseAdmin)
```

Dương Hữu Thành

85

ModelAdmin

Change course

Các công nghệ lập trình hiện đại

Image: Currently: courses/2021/06/Picture1.png Clear
Change: No file chosen

Active

Subject: Các công nghệ lập trình hiện đại

Description: Các công nghệ lập trình hiện đại

Category: Công nghệ Thông tin

Avatar: 

Dương Hữu Thành

86

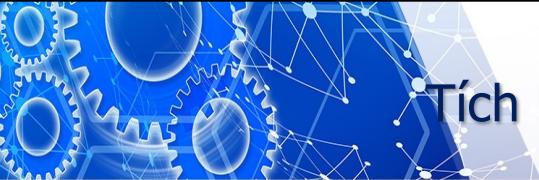
86



ModelAdmin

- Thêm CSS và JS vào trang ModelAdmin

```
class LessonAdmin(admin.ModelAdmin):  
    ...  
  
    class Media:  
        css = {  
            'all': ('/static/css/style.css', )  
        }  
        js = ('/static/js/script.js', )
```



Tích hợp CKEditor vào admin

- Cài đặt

```
pip install django-ckeditor
```

- Cập nhật biến INSTALLED_APP

```
INSTALLED_APPS = [  
    ...  
    'ckeditor',  
    'ckeditor_uploader'  
]
```

- Chi tiết: <https://django-ckeditor.readthedocs.io/>

Tích hợp CKEditor vào admin

- Thêm biến cấu hình chỉ định nơi upload

```
CKEDITOR_UPLOAD_PATH = "images/lessons/"
```

- Mặc định django sẽ upload tài nguyên vào thư mục này bên trong MEDIA_ROOT.
- Cập nhật urls của project

```
urlpatterns = [
    ...
    re_path(r'^ckeditor/',
            include('ckeditor_uploader.urls')),
]
```

- Chỉnh sửa trường content của model Lesson

```
from ckeditor.fields import RichTextField

class Lesson(ModelBase):
    ...
    content = RichTextField()
```

- Thực thi các lệnh makemigrations và migrate để cập nhật cơ sở dữ liệu.

Tích hợp CKEditor vào admin

- Cập nhật LessonAdmin trong admin.py

```
from django import forms
from ckeditor_uploader.widgets \ 
    import CKEditorUploadingWidget

class LessonForm(forms.ModelForm):
    content = forms.CharField(widget=CKEditorUploadingWidget)

    class Meta:
        model = Lesson
        fields = '__all__'

class LessonAdmin(admin.ModelAdmin):
    form = LessonForm
```

Dương Hữu Thành

91

91

Tích hợp CKEditor vào admin

- Truy cập trang thêm/cập nhật lesson của admin.

Add lesson

Image: No file chosen

Active

Subject:

Content:

Styles Format | **B** *I* U **S**

Course:

Dương Hữu Thành

92

92



InlineModelAdmin

- Các InlineModelAdmin cho phép chỉnh sửa nhiều model có quan hệ với nhau trong cùng một trang của model cha.
- Một số lớp dạng này
 - InlineModelAdmin
 - TabularInline
 - StackedInline

Dương Hữu Thành

93

93



InlineModelAdmin

- Ví dụ quan hệ ForeignKey của Lesson có một khoá ngoại đến Course

```
class LessonInlineAdmin(admin.StackedInline):  
    model = Lesson  
    fk_name = 'course' # tên khoá ngoại (tùy chọn)  
  
class CourseAdmin(admin.ModelAdmin):  
    ...  
    inlines = [LessonInlineAdmin, ]
```

Dương Hữu Thành

94

94

InlineModelAdmin

Trang cập nhật course “Các công nghệ lập trình hiện đại”

Các công nghệ lập trình hiện đại

Image: Currently: static/courses/default.jpeg Clear Change: Choose File No file chosen

Active

Subject: Các công nghệ lập trình hiện đại

Description: Các công nghệ lập trình hiện đại

Category: Công nghệ phần mềm

Avatar:

LESSONS

Lesson: Giới thiệu

Image: Currently: static/courses/default.jpeg Clear Change: Choose File No file chosen

Active

Subject: Giới thiệu

Content: Giới thiệu các công nghệ lập trình hiện đại

Dương Hữu Thành

95

95

InlineModelAdmin

- Ví dụ quan hệ ManyToMany của Tag và Lesson

```
class LessonTagInlineAdmin(admin.TabularInline):
    model = Lesson.tags.through

class TagAdmin(admin.ModelAdmin):
    inlines = [LessonTagInlineAdmin, ]

class LessonAdmin(admin.ModelAdmin):
    inlines = [LessonTagInlineAdmin, ]
```

Dương Hữu Thành

96

96

Chỉnh sửa template admin

- Tất cả tập tin template của admin tại
 - django/contrib/admin/templates/admin
- Trong thư mục templates tạo admin/app_name, để đè lên các trang admin đã có, ta có thể copy file template tương ứng vào thư mục này và tiến hành chỉnh sửa mong muốn hoặc thực hiện extends template và chỉ ghi đè những thứ cần thiết.

Dương Hữu Thành

97

97

AdminSite

- Các trang quản trị của django là thể hiện của AdminSite, ta có thể đăng ký các model và ModelAdmin với nó. Ta cũng có thể ghi đè admin site mặc định.
- Trong admin.py chỉnh sửa như sau:

```
class CourseAppAdminSite(admin.AdminSite):  
    site_header = 'Hệ thống khoá học trực tuyến'  
  
    admin_site = CourseAppAdminSite(name='myadmin')  
  
    admin_site.register(Category)  
    admin_site.register(Course, CourseAdmin)  
    admin_site.register(Lesson, LessonAdmin)  
    admin_site.register(Tag, TagAdmin)
```

Dương Hữu Thành

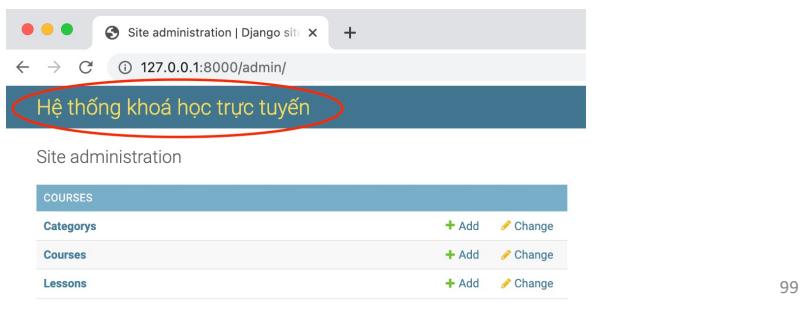
98

98

AdminSite

- Trong urls.py của project sửa url của admin

```
from courses.admin import admin_site
urlpatterns = [
    ...
    path('admin/', admin_site.urls),
    ...
]
```



Dương Hữu Thành

99

AdminSite

- Ta cũng có thể thêm view mới vào admin site.
- AdminSite cung cấp phương thức `get_urls()` để ghi đè định nghĩa thêm các view cho admin.

```
class CourseAppAdminSite(admin.AdminSite):
    def get_urls(self):
        return [
            path('course-stats/', self.stats_view)
        ] + super().get_urls()

    def stats_view(self, request):
        ...
```

Dương Hữu Thành

100

100

- Phương thức stats_view

```
class CourseAppAdminSite(admin.AdminSite):  
    ...  
    def stats_view(self, request):  
        count = Course.objects.filter(active=True).count()  
        stats = Course.objects\  
            .annotate(lesson_count=Count('my_lesson'))\  
            .values('id', 'subject', 'lesson_count')  
  
        return TemplateResponse(request,  
            'admin/course-stats.html', {  
                'course_count': count,  
                'course_stats': stats  
            })
```

- Thêm course-stats.html vào templates/admin

```
{% extends 'admin/base_site.html' %}  
{% block content %}  
<h1>THÔNG KÊ THÔNG TIN CÁC KHOÁ HỌC</h1>  
<h2>Số lượng khóa học: {{ course_count }}</h2>  
<ul>  
    {% for c in course_stats %}  
        <li><strong>{{ c.subject }}</strong> có {{  
            c.lesson_count }} bài học </li>  
    {% endfor %}  
</ul>  
{% endblock %}
```

The screenshot shows a browser window titled "Django site admin" with the URL "127.0.0.1:8000/admin/course-stats/". The page title is "Django administration". Below it is a section titled "THỐNG KÊ THÔNG TIN CÁC KHOÁ HỌC". A bold heading "Số lượng khoá học: 8" is followed by a bulleted list of categories and their counts:

- Các công nghệ lập trình hiện đại có 4 bài học
- Lập trình Java có 0 bài học
- Thiết kế Web có 0 bài học
- Kiểm thử phần mềm có 0 bài học
- Công nghệ phần mềm có 0 bài học
- Cấu trúc dữ liệu và giải thuật có 0 bài học
- Kỹ thuật lập trình có 0 bài học
- Cơ sở lập trình có 0 bài học

Dương Hữu Thành

103

The screenshot shows a slide with the title "URL Dispatcher". Below the title is a bulleted list of points explaining the URL dispatcher mechanism in Django:

- Để thiết kế URL cho app, ta tạo một module python chứa các định nghĩa ánh xạ giữa URL path và view của django.
- Django dựa trên biến ROOT_URLCONF trong settings.py để biết module chứa URLconf gốc. Tuy nhiên, nếu đối tượng HttpRequest có thuộc tính urlconf thì giá trị này được sử dụng thay cho ROOT_URLCONF.

Dương Hữu Thành

104



URL Dispatcher

- Django tìm trong các biến `urlpatterns` trong các python module.
- Django so khớp với từng URL pattern trong biến đó và chọn cái đầu tiên khớp request. Sau đó django gọi view tương ứng.
- Nếu không tìm thấy URL pattern nào khớp thì django sẽ ném ngoại lệ hoặc thực thi view của error handling.

Dương Hữu Thành

105

105



URL Dispatcher

- Ví dụ sử dụng cú pháp path converter

```
from django.urls import path
from . import views

urlpatterns = [
    path('lessons/', views.index),
    path('lessons/<int:lesson_id>', views.details)
]
```

- Converter `<int:lesson_id>` tham số yêu cầu là một số nguyên, mặc định là chuỗi.

Dương Hữu Thành

106

106



URL Dispatcher

- Một số loại converter
 - str: khớp chuỗi khác rỗng, trừ “/”.
 - int: khớp với số nguyên.
 - slug: khớp với chuỗi slug chứa các ký tự ASCII, ký số, dấu gạch chân, dấu gạch giữa.
 - uuid: khớp chuỗi định dạng UUID, trả về thẻ hiện UUID.
 - path: khớp chuỗi khác rỗng, bao gồm “/”.

Dương Hữu Thành

107



URL Dispatcher

- Ví dụ sử dụng biểu thức chính quy

```
from django.urls import path, re_path
from . import views

urlpatterns = [
    re_path(r'^courses/(?P<course_id>[0-9]{1,2})/$', views.course_detail)
]
```

Dương Hữu Thành

108



View

- View function là hàm python nhận tham số là request và trả về response. View chứa các logic xử lý cần thiết trước khi trả về response.

```
def details(request, lesson_id):  
    try:  
        lesson = Lesson.objects.get(pk=lesson_id)  
    except Lesson.DoesNotExist:  
        return HttpResponseRedirect("/lessons/not-found")  
  
    return render(request, 'lessons/detail.html',  
                  {'lesson': lesson})
```

Dương Hữu Thành

109

109



View

- Kiểm tra phương thức HTTP request trong View

```
def test(request):  
    if request.method == 'GET':  
        pass  
    elif request.method == 'POST':  
        pass  
    ...
```

Dương Hữu Thành

110

110



View

- Django cho phép tạo view bằng class, nó giúp tổ chức dễ dàng hơn theo các phương thức HTTP request.

```
from django.views import View

class CourseView(View):
    def get(self, request):
        pass

    def post(self, request):
        pass
```

Dương Hữu Thành

111

111



View

- Khi đó URLConfig cấu hình như sau để ánh xạ URL vào view.

```
from django.urls import path
from .views import CourseView

urlpatterns = [
    path('courses/', CourseView.as_view())
]
```

Dương Hữu Thành

112

112

Truy vấn dữ liệu nâng cao

- Sử dụng **annotate()** cung cấp các biểu thức truy vấn trên một trường của model, biểu thức có thể là giá trị đơn giản, biểu thức thông kê (aggregate expression: avg, sum).

```
from django.db.models import Count

c = Course.objects.annotate(Count('my_lesson'))
print(c[0].my_lesson__count)

c = Course.objects.annotate(num=Count('my_lesson'))
print(c[0].num)
```

Dương Hữu Thành

113

113

Truy vấn dữ liệu nâng cao

- values():** dùng chỉ định các trường dữ liệu muốn lấy. Phương thức trả về QuerySet, mỗi phần tử là **từ điển** có key là các giá trị chỉ định trong values().

```
Lessons = Lesson.objects \
    .values('id', 'subject',
            'course_subject') \
    .filter(active=True)

print(lessons)
<QuerySet [{{'id': 1, 'subject': 'Giới thiệu',
    'course_subject': 'Các công nghệ lập trình
    hiện đại'}}]>
```

Dương Hữu Thành

114

114

Truy vấn dữ liệu nâng cao

- Kết hợp annotate, values tạo truy vấn group by.

```
re = Category.objects\  
    .annotate(count=Count('course_id'))\  
    .values("id", "name", "count")\  
    .order_by('-id')  
  
print(re.query)  
SELECT `courses_category`.`id`,  
`courses_category`.`name`,  
COUNT(`courses_course`.`id`) AS `count`  
FROM `courses_category` LEFT OUTER JOIN  
`courses_course` ON (`courses_category`.`id` =  
`courses_course`.`category_id`)  
GROUP BY `courses_category`.`id`  
ORDER BY `courses_category`.`id` DESC
```

Dương Hữu Thành

115

Truy vấn dữ liệu nâng cao

- values_list()**: tương tự values() nhưng kết quả trả ra là QuerySet chứa các tuple, mỗi tuple tương ứng là các giá trị của các trường chỉ định trong values_list().

```
lessons = Lesson.objects\  
    .values_list('id', 'subject',  
                'course_subject')\  
    .filter(active=True)  
  
print(lessons)  
<QuerySet [(1, 'Giới thiệu', 'Các công nghệ lập  
trình hiện đại')]>
```

Dương Hữu Thành

116

116

Truy vấn dữ liệu nâng cao

- **extra()**: cho phép thêm mệnh đề vào câu truy vấn SQL được tạo ra từ QuerySet.

```
User.objects.extra(select={  
    'full_name': 'first_name + last_name'  
}).filter(is_active=True)
```

```
SELECT (first_name + last_name) AS `full_name`,  
`auth_user`.`id`, `auth_user`.`password`,  
`auth_user`.`last_login`, `auth_user`.`is_superuser`,  
`auth_user`.`username`, `auth_user`.`first_name`,  
`auth_user`.`last_name`, `auth_user`.`email`,  
`auth_user`.`is_staff`, `auth_user`.`is_active`,  
`auth_user`.`date_joined`  
FROM `auth_user` WHERE `auth_user`.`is_active`
```

Dương Hữu Thành

117

- ## Truy vấn dữ liệu nâng cao
- Sử dụng mệnh đề where trong extra().

```
User.objects\  
    .extra(where=['is_active or year(date_joined)=2021'])\  
    .filter(is_superuser=True)
```

```
SELECT `auth_user`.`id`, `auth_user`.`password`,  
`auth_user`.`last_login`, `auth_user`.`is_superuser`,  
`auth_user`.`username`, `auth_user`.`first_name`,  
`auth_user`.`last_name`, `auth_user`.`email`,  
`auth_user`.`is_staff`, `auth_user`.`is_active`,  
`auth_user`.`date_joined` FROM `auth_user`  
WHERE ((is_active or year(date_joined)=2021) AND  
`auth_user`.`is_superuser`)
```

Dương Hữu Thành

118

118

Truy vấn dữ liệu nâng cao

- Sắp xếp theo trường mới tạo ra từ extra().

```
Course.objects\  
    .extra(select={  
        'lesson_count': '''  
            Select Count(*) From courses_lesson  
            Where courses_lesson.course_id=courses_course.id  
        '''})\  
    .extra(order_by=['-lesson_count']).all()
```

Truy vấn dữ liệu nâng cao

- distinct()**: trả về QuerySet sử dụng SELECT DISTINCT trong truy vấn SQL không có các dòng trùng nhau.
- defer()**: chỉ định các trường không muốn truy vấn dữ liệu.
- only()**: ngược lại với defer(), dùng chỉ định các trường muốn truy vấn.

Truy vấn dữ liệu nâng cao

- `select_related()`: phương thức này sẽ **truy vấn sẵn** cho các quan hệ ForeignKey tham chiếu đến. Sau đó khi sử dụng khoá ngoại sẽ không cần truy vấn nữa.

```
Lesson.objects.select_related('course').all()
```

- Câu truy vấn sinh ra có dạng

```
SELECT `courses_lesson`.*  
FROM `courses_lesson` INNER JOIN `courses_course`  
    ON (`courses_lesson`.`course_id`  
        = `courses_course`.`id`)
```

Truy vấn dữ liệu nâng cao

- `prefetch_related()`: phương thức này truy vấn sẵn cho các quan hệ ForeignKey tham chiếu đến, nhưng thực hiện trên các **truy vấn riêng biệt**.

```
>>> c = Course.objects.prefetch_related('lessons').last()  
>>> c  
<Course: Các công nghệ lập trình hiện đại>  
>>> c.lessons.all()  
<QuerySet [<Lesson: Giới thiệu>, <Lesson: Phát triển API  
với Django REST framework>, <Lesson: ReactJS>]>
```

Truy vấn dữ liệu nâng cao

- **Prefetch()**: sử dụng điều khiển hoạt động của prefetch_related() để truy vấn trên quan hệ liên quan.

```
>>> from django.db.models import Prefetch
>>> c =
Course.objects.prefetch_related(Prefetch('lessons',
Lesson.objects.filter(subject_icontains='api'))).last()
>>> c
<Course: Các công nghệ lập trình hiện đại>
>>> c.lessons.all()
<QuerySet [<Lesson: Phát triển API với Django REST
framework>]>
```

Dương Hữu Thành

123

123

Truy vấn dữ liệu nâng cao

- **raw()**: thực thi câu truy vấn SQL và trả về thể hiện của django.db.models.query.RawQuerySet.

```
re = Course.objects.raw('''
    SELECT * FROM courses_course WHERE active
''')
for r in re:
    print('%d - %s' % (r.id, r.subject))
```

Dương Hữu Thành

124

124



Django Authentication

- Django authentication hỗ trợ xử lý cả authentication và authorization.
 - **authentication**: xác thực user là ai.
 - **authorization**: user đã xác thực có quyền gì.
- Django authentication tập trung trong module **django.contrib.auth** và cấu hình cần thiết đã được **thiết lập sẵn** trong project khi thực hiện lệnh `django-admin startproject`.



Django Authentication

- Biển `INSTALLED_APPS` của `settings.py`
 - `django.contrib.auth`
 - `django.contrib.contenttypes`
- Biển `MIDDLEWARE` của `settings.py`
 - `SessionMiddleware`
 - `AuthenticationMiddleware`
- Khi thực thi lệnh `migrate` sẽ tạo các bảng dữ liệu cho chứng thực và phân quyền.

Django Authentication

- User là model đại diện thông tin người dùng tương tác với hệ thống.
- Các thuộc tính chính có sẵn trong User
 - username
 - password
 - email
 - first_name
 - last_name
 - is_active, is_staff, is_superuser

Dương Hữu Thành

127

127

Django Authentication

- Tạo user

```
User.objects.create_user(  
    first_name="Thanh",  
    last_name="Duong",  
    email="dhthanhqa@gmail.com",  
    username="dhthanh",  
    password="thanh@123"  
)
```

Dương Hữu Thành

128

128

Django Authentication

- Đổi mật khẩu

```
>>> user = User.objects.get(pk=2)
>>> user.password
'pbkdf2_sha256$260000$BGIhi3YszA5OCcSBlkzqxe$n4iM
LUYQ0nmRnF6T6M/SbhU8Gxaa7l5uzCXO6F9IEoM='
>>> user.set_password('123456')
>>> user.save()
>>> user.password
'pbkdf2_sha256$260000$qWkP1k08NiWBKHZCJXNlgl$moCa
hftG7+B76rbKI7hybtF0MMRen5s50RQ+juFTCSQ='
```

- Chứng thực user

```
>>> from django.contrib.auth import authenticate
>>> authenticate(username='dhthanh', password='thanh@123')
>>> authenticate(username='dhthanh', password='123456')
<User: dhthanh>
```



Django Authentication

- Mở rộng auth.User mặc định.
 - Nếu chỉ cần đổi một số phương thức có sẵn, mà không thêm các thuộc tính lưu trữ thì ta có thể sử dụng một proxy model kế thừa User model.
 - Ngược lại, ta có thể sử dụng quan hệ OneToOne giữa User và một model chứa thêm thông tin bổ sung cho User.

Dương Hữu Thành

131



Django Authentication

- Ngoài ra, khi bắt đầu project, ta có thể tạo một custom user model sẵn thay cho model user mặc định, khi có nhu cầu bổ sung thông tin trong tương lai sẽ dễ dàng hơn.

```
from django.contrib.auth.models import AbstractUser

class User(AbstractUser):
    pass
```

- Chú ý nên cấu hình lại trong settings.py thuộc tính AUTH_USER_MODEL = "myapp.MyUser"

Dương Hữu Thành

132

132

Django Authentication

- Giả sử ta muốn user có thêm thông tin avatar và sử dụng lớp User định nghĩa để chứng thực.

```
from django.contrib.auth.models import AbstractUser

class User(AbstractUser):
    avatar = models.ImageField(upload_to="users/%Y/%m/")
```

- Cấu hình thuộc tính AUTH_USER_MODEL trong settings.py

```
AUTH_USER_MODEL = 'courses.User'
```

Dương Hữu Thành

133

- Chạy các lệnh makemigrations và migrate để tạo cơ sở dữ liệu, ta sẽ được table như sau

courses_user	
	Columns
◆	id
◆	password
◆	last_login
◆	is_superuser
◆	username
◆	first_name
◆	last_name
◆	email
◆	is_staff
◆	is_active
◆	date_joined
◆	avatar
>	Indexes
>	Foreign Keys
>	Triggers

Dương Hữu Thành

134

134



Phân quyền

- Django cũng cung cấp sẵn cơ chế phân quyền cho user hoặc một nhóm user.
- Lớp model User có hai quan hệ Many To Many
 - groups: các nhóm user thuộc vào
 - user_permissions: các quyền user được phép
- Từ hai quan hệ này, ta có thể thêm, cập nhật, xoá quyền user được phép thực hiện.
- Ta có thể gom nhóm các user là thể hiện của django.contrib.auth.models.Group, user trong group sẽ có đầy đủ quyền của group đó.

Dương Hữu Thành

135



Phân quyền

- Khi django.contrib.auth trong INSTALLED_APP đảm bảo được 4 quyền mặc định (add, change, delete, view) được tạo cho mỗi model trong các app của project.
- Những quyền này sẽ được tạo khi thực thi lệnh manage.py migrate cho mỗi model mới.

Dương Hữu Thành

136

Phân quyền

- Giả sử ta có app với app_label=courses và có model tên Lesson, ta có thể kiểm tra các permission đối tượng user được phép thực hiện như sau:
 - add: user.has_perm(courses.add_lesson)
 - change: user.has_perm(courses.change_lesson)
 - delete: user.has_perm(courses.delete_lesson)
 - view: user.has_perm(courses.view_bar)

Dương Hữu Thành

137

137

Phân quyền

The screenshot shows the Django Admin interface for managing user permissions. On the left, there's a sidebar with links like 'Groups', 'Users', 'COURSES' (with 'Category', 'Courses', 'Lessons'), 'DJANGO OAUTH TOOLKIT' (with 'Access tokens', 'Applications', 'Grants', 'Id tokens', 'Refresh tokens'), and 'Important dates'. The main content area is titled 'User permissions' and contains two sections: 'Available user permissions' and 'Chosen user permissions'. A red circle highlights the 'Available user permissions' section, which lists numerous permissions such as 'contenttypes | content type | Can view content type', 'courses | category | Can add category', etc. Below this section is a note: 'Các permission mặc định được thêm cho các model: category, course, lesson'. At the bottom of the page, there are fields for 'Last login' (Date: 2021-06-02, Time: 02:15:36) and 'Date joined' (Date: 2021-05-29, Time: 03:09:28), along with buttons for 'Delete', 'Save and add another', 'Save and continue editing', and 'SAVE'.

Dương Hữu Thành

138

138

Phân quyền

- Ngoài ra, ta cũng có thể thêm một permission mới thể thiết lập cho user hoặc group.

```
from courses.models import Lesson
from django.contrib.auth.models import Permission
from django.contrib.contenttypes.models import ContentType

content_type = ContentType.objects.get_for_model(Lesson)
per = Permission.objects\ 
    .create(codename='can_add_tags_to_lesson',
            name='Can add tags to a lesson',
            content_type=content_type)
```

Dương Hữu Thành

139

139

Chứng thực trong Request

- Django sử dụng session và middleware để gắn hệ thống chứng thực vào các đối tượng request.
- Thuộc tính **request.user** của mỗi request là thể hiện User đại diện cho user hiện hành đã chứng thực, nếu user chưa chứng thực thì nó là thể hiện của AnonymousUser.
- Sử dụng **request.user.is_authenticated** kiểm tra user đã được chứng thực chưa.

Dương Hữu Thành

140

140

Chứng thực trong Request

```
from django.contrib.auth import authentication, login, logout

def login(request):
    username = request.POST['username']
    password = request.POST['password']
    user = authenticate(request,
                        username=username,
                        password=password)
    if user is not None:
        login(request, user)
        ...
    else:
        ...

def logout_view(request):
    logout(request)
```

Dương Hữu Thành

141

141

Chứng thực trong Request

- Bắt buộc phải chứng thực user sử dụng decorator cho các function view
 - @login_required(redirect_field_name='next', login_url=settings.LOGIN_URL)
- Kế thừa lớp LoginRequiredMixin cho các class view bắt buộc chứng thực với hai thuộc tính tương ứng như login_required trên là login_url và redirect_field_name.
 - django.contrib.auth.mixins.LoginRequiredMixin

Dương Hữu Thành

142

142



Chứng thực trong Request

- Giới hạn quyền truy cập cho user phải thoả một điều kiện nào đó, sử dụng decorator cho function view.
 - `@user_passes_test(test_func, login_url=settings.LOGIN_URL, redirect_field_name='next')`
 - Hàm `test_func` có 1 đối số là đối tượng user hiện hành.
- Sử dụng `UserPassesTestMixin` cho class view.

Dương Hữu Thành

143

143



Chứng thực trong Request

- Sử dụng decorator cho các function view để kiểm tra có được quyền truy cập
 - `@permission_required(perm='app_label.permission_codename', login_url=settings.LOGIN_URL, raise_exception=False)`
- Sử dụng lớp `PermissionRequiredMixin` cho class view.

Dương Hữu Thành

144

144



Q&A

ThS. Dương Hữu Thành,
Khoa CNTT, Đại học Mở TP.HCM,
[thanh.dh@ou.edu.vn.](mailto:thanh.dh@ou.edu.vn)

145