免費電子書

學習

Django

Free unaffiliated eBook created from
Stack Overflow contributors.

#django

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: django

It is an unofficial and free Django ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official Django.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

1: Django

Django“”“DjangoWeb”。 MVC。

- Web
- HTML
-
-
-
- Django
- DjangoXML/JSON
-
- Python

1.11	201744
1.10	201681
1.9	2015121
1.8	2015-04-01
1.7	201492
1.6	2013116
1.5	2013226
1.4	2012-03-23
1.3	2011-03-23
1.2	2010-05-17
1.1	2009-07-29
1.0	2008-09-03

Examples

DjangoPythonWeb。 Django**1.11**Python **2.73.43.53.6**。 `pip`。 django

```
$ pip install django
```

djangodjango **1.10.5**

```
$ pip install django==1.10.5
```

DjangoWebDjango。 `django-admin`

```
$ django-admin startproject myproject
```

`myproject` 。

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

```
$ cd myproject
$ python manage.py runserver
```

Web`http://127.0.0.1:8000/` 。

`runserver8000`IP。 。 。

。

```
$ python manage.py runserver 8080
```

IP。

```
$ python manage.py runserver 0.0.0.0:8000
```

`runserver`。 Apache。

Django

Django`apps` 。 。 `manage.pystartapp`*myapp*

```
python manage.py startapp myapp
```

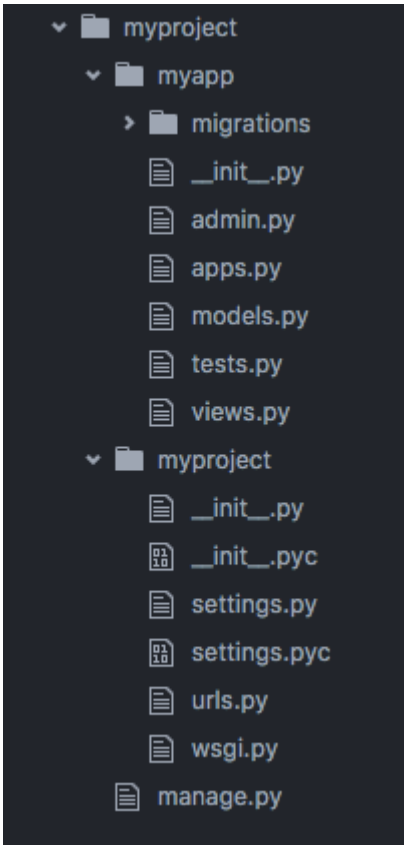
myapp`models.pyviews.py` ◦

Django*myapp* `settings.py`

```
# myproject/settings.py

# Application definition
INSTALLED_APPS = [
    ...
    'myapp',
]
```

Django◦ `/src`◦



Django

django-admin Django◦ Django◦ `./manage.py`◦ `DJANGO_SETTINGS_MODULE`◦

DjangoDjangoPython◦ Django`django-admin startproject NAME`◦ `manage.pyurls.py`URL◦ `manage.py django-admin`◦ `python manage.py runserver` ◦ Django◦

DjangoPython`models.py` URL◦ `django-admin startapp NAME`◦ `settings.py`INSTALLED_APPS◦ Django◦ Django◦

Django ORM`models.py`◦ `settings.py`DATABASES`settings.py` DATABASES ◦ `python manage.py makemigrationspython manage.py makemigrations python manage.py migrate` ◦

hello world。

1Django。

```
pip install Django
```

2

```
django-admin startproject hello
```

`hello`

```
hello/
├── hello/
│   ├── __init__.py
│   ├── settings.py
│   ├── urls.py
│   └── wsgi.py
└── manage.py
```

3`hello__init.py__views.py`

```
hello/
├── hello/
│   ├── __init__.py
│   ├── settings.py
│   ├── urls.py
│   ├── views.py  <- here
│   └── wsgi.py
└── manage.py
```

```
from django.http import HttpResponse

def hello(request):
    return HttpResponse('Hello, World')
```

。

4`hello/urls.py`

```
from django.conf.urls import url
from django.contrib import admin
from hello import views

urlpatterns = [
    url(r'^admin/', admin.site.urls),
    url(r'^$', views.hello)
]
```

`hello()`URL。

5。

```
python manage.py runserver
```

6

`http://localhost:8000/`

“”。 PythonPython。

DjangoPython。

Python 3.3+

Python 3.3+`venv`pyvenv。 `pyvenv`python3 -m venv。

```
$ pyvenv <env-folder>
# Or, if pyvenv is not available
$ python3 -m venv <env-folder>
```

Python 2

Python 2pip

```
$ pip install virtualenv
```

`virtualenv`

```
$ virtualenv <env-folder>
```

。 。

“”Python

Linux

```
$ source <env-folder>/bin/activate
```

Windows

```
<env-folder>\Scripts\activate.bat
```

。 `(<env-folder>) $`

`pip`。

`deactivate`

```
(<env-folder>) $ deactivate
```

virtualenvwrapper

virtualenvwrapper virtualenv

```
# Create a virtualenv
mkvirtualenv my_virtualenv

# Activate a virtualenv
workon my_virtualenv

# Deactivate the current virtualenv
deactivate
```

pyenv + pyenv-viritualenv

Pythonvirtualenvpyenv-virtualenv

```
# Create a virtualenv for specific Python version
pyenv virtualenv 2.7.10 my-virtual-env-2.7.10

# Create a vritualenv for active python verion
pyenv virtualenv venv34

# Activate, deactivate virtualenv
pyenv activate <name>
pyenv deactivate
```

virtualenvspostactivatePYTHONPATHDJANGO_SETTINGS_MODULE。

```
#!/bin/sh
# This hook is sourced after this virtualenv is activated

# Set PYTHONPATH to isolate the virtualenv so that only modules installed
# in the virtualenv are available
export PYTHONPATH="/home/me/path/to/your/project_root:$VIRTUAL_ENV/lib/python3.4"

# Set DJANGO_SETTINGS_MODULE if you don't use the default `myproject.settings`
# or if you use `django-admin` rather than `manage.py`
export DJANGO_SETTINGS_MODULE="myproject.settings.dev"
```

<env-folder>.project。 。

<env-folder>/.project。 。

```
/path/to/project/directory
```

source <env-folder>/bin/activateworkon my_virtualenv /path/to/project/directory 。

Hello World

DjangoHello World。 django-admin startproject example。

1. file.py。
2. 。

```
import sys

from django.conf import settings

settings.configure(
    DEBUG=True,
    SECRET_KEY='thisisthesecretkey',
    ROOT_URLCONF=__name__,
    MIDDLEWARE_CLASSES=(
        'django.middleware.common.CommonMiddleware',
        'django.middleware.csrf.CsrfViewMiddleware',
        'django.middleware.clickjacking.XFrameOptionsMiddleware',
    ),
)

from django.conf.urls import url
from django.http import HttpResponse

# Your code goes below this line.

def index(request):
    return HttpResponse('Hello, World!')

urlpatterns = [
    url(r'^$', index),
]

# Your code goes above this line

if __name__ == "__main__":
    from django.core.management import execute_from_command_line

    execute_from_command_line(sys.argv)
```

3. python file.py runserver。
4. 127.0.0.1:8000 。

Docker。

Djangodevops。 。

GitHubDjango。

```
PROJECT_ROOT
├── devel.dockerfile
├── docker-compose.yml
├── nginx
│   └── project_name.conf
├── README.md
├── setup.py
└── src
    ├── manage.py
    └── project_name
        ├── __init__.py
        └── service
            ├── __init__.py
            ├── settings
            │   ├── common.py
            │   ├── development.py
            │   ├── __init__.py
            │   └── staging.py
            ├── urls.py
            └── wsgi.py
```

serviceserviceDockerfile ◦

Dockerfile

Docker◦ `devel.dockerfile`

```
FROM python:2.7
ENV PYTHONUNBUFFERED 1

RUN mkdir /run/service
ADD . /run/service
WORKDIR /run/service

RUN pip install -U pip
RUN pip install -I -e .[develop] --process-dependency-links

WORKDIR /run/service/src
ENTRYPOINT ["python", "manage.py"]
CMD ["runserver", "0.0.0.0:8000"]
```

Docker - ◦

Docker - ◦ `docker-compose.yml`

```
version: '2'
services:
  web:
    build:
      context: .
      dockerfile: devel.dockerfile
    volumes:
```

```
      - "./src/{{ project_name }}:/run/service/src/{{ project_name }}"
      - "./media:/run/service/media"
    ports:
      - "8000:8000"
    depends_on:
      - db
  db:
    image: mysql:5.6
    environment:
      - MYSQL_ROOT_PASSWORD=root
      - MYSQL_DATABASE={{ project_name }}
  nginx:
    image: nginx
    ports:
      - "80:80"
    volumes:
      - "./nginx:/etc/nginx/conf.d"
      - "./media:/var/media"
    depends_on:
      - web
```

Nginx

prodNginx。 nginx

```
server {
    listen   80;
    client_max_body_size 4G;
    keepalive_timeout 5;

    location /media/ {
        autoindex on;
        alias /var/media/;
    }

    location / {
        proxy_pass_header Server;
        proxy_set_header Host $http_host;
        proxy_redirect off;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Scheme $scheme;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Ssl on;
        proxy_connect_timeout 600;
        proxy_read_timeout 600;
        proxy_pass http://web:8000/;
    }
}
```

```
$ cd PROJECT_ROOT
$ docker-compose build web  # build the image - first-time and after requirements change
$ docker-compose up  # to run the project
$ docker-compose run --rm --service-ports --no-deps  # to run the project - and be able to use
PDB
$ docker-compose run --rm --no-deps <management_command>  # to use other than runserver
```

```
commands, like makemigrations
$ docker exec -ti web bash  # For accessing django container shell, using it you will be
inside /run/service directory, where you can run ./manage shell, or other stuff
$ docker-compose start  # Starting docker containers
$ docker-compose stop  # Stopping docker containers
```

Django https://riptutorial.com/zh-TW/django/topic/200/django

2: ArrayField - PostgreSQL

- django.contrib.postgres.fields import ArrayField
- class ArrayFieldbase_fieldsize = None** options
- FooModel.objects.filterarray_field_name__contains = [objectstocheck]
- FooModel.objects.filterarray_field_name__contained_by = [objectstocheck]

`size`PostgreSQLPostgreSQL。

`ArrayField`Postgresql。

;。 。 。

Examples

ArrayField

PostgreSQL ArrayFieldArrayField。 `FloatField` 。

```
from django.db import models, FloatField
from django.contrib.postgres.fields import ArrayField

class Book(models.Model):
    ratings = ArrayField(FloatField())
```

ArrayField

```
from django.db import models, IntegerField
from django.contrib.postgres.fields import ArrayField

class IceCream(models.Model):
    scoops = ArrayField(IntegerField()  # we'll use numbers to ID the scoops
                  , size=6)  # our parlor only lets you have 6 scoops
```

sizepostgresqlpostgresql`scoops`7。

containsArrayField

。

```
VANILLA, CHOCOLATE, MINT, STRAWBERRY = 1, 2, 3, 4  # constants for flavors
choco_vanilla_cones = IceCream.objects.filter(scoops__contains=[CHOCOLATE, VANILLA])
```

`models.pyIceCream`。

django`ArrayField`。 RunSQL。

ArrayFields

ArrayFieldArrayField base_field ◦

```
from django.db import models, IntegerField
from django.contrib.postgres.fields import ArrayField

class SudokuBoard(models.Model):
    numbers = ArrayField(
        ArrayField(
            models.IntegerField(),
            size=9,
        ),
        size=9,
    )
```

contains_by

◦

```
minty_vanilla_cones = IceCream.objects.filter(scoops__contained_by=[MINT, VANILLA])
```

ArrayField - PostgreSQL https://riptutorial.com/zh-TW/django/topic/1693/arrayfield-----postgresql

3: Django Rest Framework

Examples

API

Django REST Framework“DRF” API。

models.py

```
class FeedItem(models.Model):
    title = models.CharField(max_length=100, blank=True)
    url = models.URLField(blank=True)
    style = models.CharField(max_length=100, blank=True)
    description = models.TextField(blank=True)
```

Django`FeedItem` JSON。 Django。 。

serializers.py

```
from rest_framework import serializers
from . import models

class FeedItemSerializer(serializers.ModelSerializer):
    class Meta:
        model = models.FeedItem
        fields = ('title', 'url', 'description', 'style')
```

views.py

DRF。 APIDRF`ListAPIView`。

。

```
from rest_framework import generics
from . import serializers, models

class FeedItemList(generics.ListAPIView):
    serializer_class = serializers.FeedItemSerializer
    queryset = models.FeedItem.objects.all()
```

urls.py

DRF。

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

urlpatterns = [
    ...
```

```
    url(r'path/to/api', views.FeedItemList.as_view()),
]
```

Django Rest Framework https://riptutorial.com/zh-TW/django/topic/7341/django-rest-framework

4: DjangoCRUD

Examples

CRUD

。Stack Overflow Documentation。

```
django-admin startproject myproject
cd myproject
python manage.py startapp myapp
```

myproject / settings.py

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

myappurls.py。

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

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

urls.py。

```
from django.conf.urls import url
from django.contrib import admin
from django.conf.urls import include
from myapp import views

urlpatterns = [
    url(r'^$', views.index, name='index'),
    url(r'^myapp/', include('myapp.urls')),
    url(r'^admin/', admin.site.urls),
]
```

myapptemplates。 **templates**index.html。 。

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

```
<head>
    <title>myapp</title>
</head>
<body>
    <h2>Simplest Crud Example</h2>
    <p>This shows a list of names and lets you Create, Update and Delete them.</p>
    <h3>Add a Name</h3>
    <button>Create</button>
</body>
</html>
```

views.pyindex.html

```
from django.shortcuts import render, redirect

# Create your views here.
def index(request):
    return render(request, 'index.html', {})
```

。 。 **models.py**。

```
from __future__ import unicode_literals

from django.db import models

# Create your models here.
class Name(models.Model):
    name_value = models.CharField(max_length=100)

    def __str__(self): # if Python 2 use __unicode__
        return self.name_value
```

Name。

```
python manage.py createsuperuser
python manage.py makemigrations
python manage.py migrate
```

Django。Django。admin。**admin.py**。

```
from django.contrib import admin
from myapp.models import Name
# Register your models here.

admin.site.register(Name)
```

`python manage.py runserver`。http// localhost8000 /。http// localhost8000 / admin。MYAPP100。

。 **views.py**Name。

```
from django.shortcuts import render, redirect
from myapp.models import Name

# Create your views here.
```

```
def index(request):
    names_from_db = Name.objects.all()
    context_dict = {'names_from_context': names_from_db}
    return render(request, 'index.html', context_dict)
```

index.html。

```
<!DOCTYPE html>
<html>
<head>
    <title>myapp</title>
</head>
<body>
    <h2>Simplest Crud Example</h2>
    <p>This shows a list of names and lets you Create, Update and Delete them.</p>
    {% if names_from_context %}
        <ul>
            {% for name in names_from_context %}
                <li>{{ name.name_value }}  <button>Delete</button>
<button>Update</button></li>
            {% endfor %}
        </ul>
    {% else %}
        <h3>Please go to the admin and add a Name under 'MYAPP'</h3>
    {% endif %}
    <h3>Add a Name</h3>
    <button>Create</button>
</body>
</html>
```

CRUD。**myapp**forms.py。

```
from django import forms
from myapp.models import Name

class NameForm(forms.ModelForm):
    name_value = forms.CharField(max_length=100, help_text = "Enter a name")

    class Meta:
        model = Name
        fields = ('name_value',)
```

index.html

```
<!DOCTYPE html>
<html>
<head>
    <title>myapp</title>
</head>
<body>
    <h2>Simplest Crud Example</h2>
    <p>This shows a list of names and lets you Create, Update and Delete them.</p>
    {% if names_from_context %}
        <ul>
            {% for name in names_from_context %}
                <li>{{ name.name_value }}  <button>Delete</button>
<button>Update</button></li>
```

```
            {% endfor %}
        </ul>
    {% else %}
        <h3>Please go to the admin and add a Name under 'MYAPP'</h3>
    {% endif %}
    <h3>Add a Name</h3>
    <form id="name_form" method="post" action="/">
        {% csrf_token %}
        {% for field in form.visible_fields %}
            {{ field.errors }}
            {{ field.help_text }}
            {{ field }}
        {% endfor %}
        <input type="submit" name="submit" value="Create">
    </form>
</body>
</html>
```

views.py

```
from django.shortcuts import render, redirect
from myapp.models import Name
from myapp.forms import NameForm

# Create your views here.
def index(request):
    names_from_db = Name.objects.all()

    form = NameForm()

    context_dict = {'names_from_context': names_from_db, 'form': form}

    if request.method == 'POST':
        form = NameForm(request.POST)

        if form.is_valid():
            form.save(commit=True)
            return render(request, 'index.html', context_dict)
        else:
            print(form.errors)

    return render(request, 'index.html', context_dict)
```

C in create completed。

TODO

DjangoCRUD https://riptutorial.com/zh-TW/django/topic/7317/djangocrud

5: Django

	Django-Allauth。
ACCOUNT_AUTHENTICATION_METHOD=“”“” “username_email”	- 。 “”ACCOUNT_EMAIL_REQUIRED = True
ACCOUNT_EMAIL_CONFIRMATION_EXPIRE_DAYS = 3	。
ACCOUNT_EMAIL_REQUIRED= False	。 `ACCOUNT_AUTHENTICATION_METHOD`
ACCOUNT_EMAIL_VERIFICATION=“”	- “”“”“”。 “”。 “”“”。 “”“”。
ACCOUNT_LOGIN_ATTEMPTS_LIMIT= 5	。 ACCOUNT_LOGIN_ATTEMPTS_TIMEOUT 。 allauthDjango。
ACCOUNT_LOGOUT_ON_PASSWORD_CHANGE=	。
SOCIALACCOUNT_PROVIDERS= dict	

Examples

python-social-auth

python-social-auth。 FacebookTwitterGithubLinkedIn

python-social-auth

```
pip install python-social-auth
```

github 。 `requirements.txt`。

settings.py

settings.py

```
INSTALLED_APPS = (
    ...
    'social.apps.django_app.default',
    ...
)
```

AUTHENTICATION_BACKENDS。

```
AUTHENTICATION_BACKENDS = (
    'social.backends.open_id.OpenIdAuth',
    'social.backends.google.GoogleOpenId',
    'social.backends.google.GoogleOAuth2',
    'social.backends.google.GoogleOAuth',
    'social.backends.twitter.TwitterOAuth',
    'social.backends.yahoo.YahooOpenId',
    ...
    'django.contrib.auth.backends.ModelBackend',
)
```

settings.pyAUTHENTICATION_BACKENDS。 。 'django.contrib.auth.backends.ModelBackend',/。

FacebookLinkedin BackendsAPI

```
SOCIAL_AUTH_FACEBOOK_KEY = 'YOURFACEBOOKKEY'
SOCIAL_AUTH_FACEBOOK_SECRET = 'YOURFACEBOOKSECRET'
```

```
SOCIAL_AUTH_LINKEDIN_KEY = 'YOURLINKEDINKEY'
SOCIAL_AUTH_LINKEDIN_SECRET = 'YOURLINKEDINSECRET'
```

FacebookLinkedinAPISecret。

。 Stack Overflow。 。

TEMPLATE_CONTEXT_PROCESSORS

```
TEMPLATE_CONTEXT_PROCESSORS = (
    ...
    'social.apps.django_app.context_processors.backends',
    'social.apps.django_app.context_processors.login_redirect',
    ...
)
```

Django 1.8TEMPLATE_CONTEXT_PREPROCESSORS 。 TEMPLATES dict。

```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR, "templates")],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
                'social.apps.django_app.context_processors.backends',
                'social.apps.django_app.context_processors.login_redirect',
            ],
        },
    },
]
```

settings.py

```
SOCIAL_AUTH_USER_MODEL = 'somepackage.models.CustomUser'
```

CustomUser。

urls.py

```
# if you haven't imported inlcude make sure you do so at the top of your file
from django.conf.urls import url, include

urlpatterns = patterns('',
    ...
    url('', include('social.apps.django_app.urls', namespace='social'))
    ...
)
```

```
./manage.py migrate
```

```
    <a href="{% url 'social:begin' 'facebook' %}?next={{ request.path }}">Login with
Facebook</a>
    <a href="{% url 'social:begin' 'linkedin' %}?next={{ request.path }}">Login with
Linkedin</a>
```

“facebook”。

。

```
<a href="{% url 'logout' %}">Logout</a>
```

```
<a href="/logout">Logout</a>
```

urls.py

```
url(r'^logout/$', views.logout, name='logout'),
```

views.py

```
def logout(request):
    auth_logout(request)
    return redirect('/')
```

Django Allauth

Django-Allauth

- 50
-
-

- -
-
-

Django-Allauth。

Django 1.10+

```
pip install django-allauth
```

settings.py

```
# Specify the context processors as follows:
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                # Already defined Django-related contexts here

                # `allauth` needs this from django. It is there by default,
                # unless you've devilishly taken it away.
                'django.template.context_processors.request',
            ],
        },
    },
]

AUTHENTICATION_BACKENDS = (
    # Needed to login by username in Django admin, regardless of `allauth`
    'django.contrib.auth.backends.ModelBackend',

    # `allauth` specific authentication methods, such as login by e-mail
    'allauth.account.auth_backends.AuthenticationBackend',
)

INSTALLED_APPS = (
# Up here is all your default installed apps from Django

# The following apps are required:
'django.contrib.auth',
'django.contrib.sites',

'allauth',
'allauth.account',
'allauth.socialaccount',

# include the providers you want to enable:
'allauth.socialaccount.providers.google',
'allauth.socialaccount.providers.facebook',
)

# Don't forget this little dude.
SITE_ID = 1
```

settings.pyurls.py。 yourapp/urls.pyProjectName/urls.py yourapp/urls.pyProjectName/urls.py

```
urlpatterns = [
    # other urls here
    url(r'^accounts/', include('allauth.urls')),
    # other urls here
]
```

`include('allauth.urls')`

```
^accounts/ ^ ^signup/$ [name='account_signup']
^accounts/ ^ ^login/$ [name='account_login']
^accounts/ ^ ^logout/$ [name='account_logout']
^accounts/ ^ ^password/change/$ [name='account_change_password']
^accounts/ ^ ^password/set/$ [name='account_set_password']
^accounts/ ^ ^inactive/$ [name='account_inactive']
^accounts/ ^ ^email/$ [name='account_email']
^accounts/ ^ ^confirm-email/$ [name='account_email_verification_sent']
^accounts/ ^ ^confirm-email/(?P<key>[-:\w]+)/$ [name='account_confirm_email']
^accounts/ ^ ^password/reset/$ [name='account_reset_password']
^accounts/ ^ ^password/reset/done/$ [name='account_reset_password_done']
^accounts/ ^ ^password/reset/key/(?P<uidb36>[0-9A-Za-z]+)-(?P<key>.+)/$
[name='account_reset_password_from_key']
^accounts/ ^ ^password/reset/key/done/$ [name='account_reset_password_from_key_done']
^accounts/ ^social/
^accounts/ ^google/
^accounts/ ^twitter/
^accounts/ ^facebook/
^accounts/ ^facebook/login/token/$ [name='facebook_login_by_token']
```

`python ./manage.py migrate`Django-allauth。

。

Django Admin `localhost:8000/admin` `Social Applications`。

。

“ ”。

Django https://riptutorial.com/zh-TW/django/topic/4743/django

6: Django

Examples

CBVdjango-filter

`django-filter`Django QuerySets。

```
from django.db import models

class Product(models.Model):
    name = models.CharField(max_length=255)
    price = models.DecimalField()
    description = models.TextField()
    release_date = models.DateField()
    manufacturer = models.ForeignKey(Manufacturer)
```

```
import django_filters

class ProductFilter(django_filters.FilterSet):
    name = django_filters.CharFilter(lookup_expr='iexact')

    class Meta:
        model = Product
        fields = ['price', 'release_date']
```

CBVListView`get_queryset()` querset

```
from django.views.generic import ListView
from .filters import ProductFilter

class ArticleListView(ListView):
    model = Product

    def get_queryset(self):
        qs = self.model.objects.all()
        product_filtered_list = ProductFilter(self.request.GET, queryset=qs)
        return product_filtered_list.qs
```

`f.qs`。 。

Django https://riptutorial.com/zh-TW/django/topic/6101/django

7: F

FDjangoPythonPython。 。

- django.db.modelsF.

Examples

。

```
article = Article.objects.get(pk=69)
article.views_count += 1
article.save()
```

views_count1337

```
UPDATE app_article SET views_count = 1338 WHERE id=69
```

HTTP`article.save()Article.objects.get(pk=69) article.save()` 。 `views_count = 1337 views_count = 13381339` 。

`F()`

```
article = Article.objects.get(pk=69)
article.views_count = F('views_count') + 1
article.save()
```

```
UPDATE app_article SET views_count = views_count + 1 WHERE id=69
```

id51 2upvotes。
PythonN N

```
for article in Article.objects.filter(author_id=51):
    article.upvotes -= 2
    article.save()
    # Note that there is a race condition here but this is not the focus
    # of this example.
```

Pythonupvotes
`F()`

```
Article.objects.filter(author_id=51).update(upvotes=F('upvotes') - 2)
```

SQL

```
UPDATE app_article SET upvotes = upvotes - 2 WHERE author_id = 51
```

-

Python。

- 。

F() + - */。

- ```
  class MyModel(models.Model):
      int_1 = models.IntegerField()
      int_2 = models.IntegerField()
  ```

- MyModel MyModel int_1int_2 int_1 + int_2 >= 5 。 annotate()filter()

  ```
  result = MyModel.objects.annotate(
               diff=F(int_1) + F(int_2)
           ).filter(diff__gte=5)
  ```

  result。

Integer。

F https://riptutorial.com/zh-TW/django/topic/2765/f--
  ```

8: JSONField - PostgreSQL

- JSONField**
- DjangoJSONFieldPostgres JSONBPostgres 9.4。
- JSONField。 。
- JSONField JSONField。

。 。

。

Examples

JSONField

Django 1.9+

```
from django.contrib.postgres.fields import JSONField
from django.db import models

class IceCream(models.Model):
    metadata = JSONField()
```

**options 。

settings.py'django.contrib.postgres'INSTALLED_APPS

JSONField

Pythonlist dict str None bool。

```
IceCream.objects.create(metadata={
    'date': '1/1/2016',
    'ordered by': 'Jon Skeet',
    'buyer': {
         'favorite flavor': 'vanilla',
         'known for': ['his rep on SO', 'writing a book']
     },
    'special requests': ['hot sauce'],
})
```

“”JSONField。

```
IceCream.objects.filter(metadata__ordered_by='Guido Van Rossum')
```

```
IceCream.objects.filter(metadata__buyer__favorite_flavor='chocolate')
```

“”。

。

```
IceCream.objects.filter(metadata__buyer__known_for__0='creating stack overflow')
```

“”。

JSONField

Django`JSONField`JSONField。 RawSQLPostgreSQLjsonb

```
from django.db.models.expressions import RawSQL
RatebookDataEntry.objects.all().order_by(RawSQL("data->>%s", ("json_objects_key",)))
```

JSONFielddatadata['json_objects_key'] JSONField

```
data = JSONField()
```

JSONField - PostgreSQL https://riptutorial.com/zh-TW/django/topic/1759/jsonfield-----postgresql

9: Meta

DjangoPython“MetaDocumentation Guidelines”。

。 。

Examples

Django。 。

。

1.6

Django <1.6。

1.8

Django <1.8。

。 1.3–1.9 。 All versions 。 。

1.8 **1** 1.9 **2** 1.10 **1**

1. Django。
2. 。

Meta https://riptutorial.com/zh-TW/django/topic/5243/meta-

10: RangeFields - PostgreSQL

- django.contrib.postgres.fields import * RangeField
- IntegerRangeField**
- BigIntegerRangeField**
- FloatRangeField**
- DateTimeRangeField**
- DateRangeField**

Examples

Python`RangeField` ◦ `IntegerField` `BigIntegerFieldFloatField` ◦ `psycopg2` `NumericRange` Python◦ ◦

```
class Book(models.Model):
    name = CharField(max_length=200)
    ratings_range = IntegerRange()
```

RangeField

1. `'django.contrib.postgres'INSTALLED_APPS`
2. `psycopg2`

Python`NumericRange`◦

```
Book.objects.create(name='Pro Git', ratings_range=(5, 5))
```

`NumericRange`

```
Book.objects.create(name='Pro Git', ratings_range=NumericRange(5, 5))
```

3◦

```
bad_books = Books.objects.filter(ratings_range__contains=(1, 3))
```

contained_by

6◦

```
all_books = Book.objects.filter(ratings_range_contained_by=(0, 6))
```

610◦

```
Appointment.objects.filter(time_span__overlap=(6, 10))
```

None

4。

```
maybe_good_books = Books.objects.filter(ratings_range__contains=(4, None))
```

```
from datetime import timedelta

from django.utils import timezone
from psycopg2.extras import DateTimeTZRange

# To create a "period" object we will use psycopg2's DateTimeTZRange
# which takes the two datetime bounds as arguments
period_start = timezone.now()
period_end = period_start + timedelta(days=1, hours=3)
period = DateTimeTZRange(start, end)

# Say Event.timeslot is a DateTimeRangeField

# Events which cover at least the whole selected period,
Event.objects.filter(timeslot__contains=period)

# Events which start and end within selected period,
Event.objects.filter(timeslot__contained_by=period)

# Events which, at least partially, take place during the selected period.
Event.objects.filter(timeslot__overlap=period)
```

RangeFields - PostgreSQL https://riptutorial.com/zh-TW/django/topic/2630/rangefields----postgresql

11: URL

Examples

Django

DjangoURL。 view。 URL。 DjangoURLURLconf。 `<myproject>/urls.py`URLconf。

URLconfpython`urlpatterns`django.conf.urls.url()。 `url()`URL URLconf。 URL。

```
# In <myproject>/urls.py

from django.conf.urls import url

from myapp.views import home, about, blog_detail

urlpatterns = [
    url(r'^$', home, name='home'),
    url(r'^about/$', about, name='about'),
    url(r'^blog/(?P<id>\d+)/$', blog_detail, name='blog-detail'),
]
```

URLconfURL `home aboutblog-detail` 。

- `url(r'^$', home, name='home'),`

'^''$'。 URL`myapp.viewshome`。

- `url(r'^about/$', about, name='about'),`

`about/`。 URL `/about/about`。 URL/Django。

- `url(r'^blog/(?P<id>\d+)/$', blog_detail, name='blog-detail'),`

`。 blog/ 。 (?P<id>\d+)`。 Django。

`(?P<name>pattern) 。 name`Django。 。

`id blog_detailid`。 `\d+ 。 \d。 +`。

`\d+`	ID	
`[\w-]+`		
`[0-9]{4}`		
`[0-9]{2}`		
`[^/]+`		

`blog-detail/`。

- `/blog/1/ # passes id='1'`
- `/blog/42/ # passes id='42'`

- `/blog/a/ # 'a' does not match '\d'`
- `/blog// # no characters in the capturing group does not match '+'`

Django`urlpatterns`URL。 URL。

```
urlpatterns = [
    url(r'blog/(?P<slug>[\w-]+)/$', blog_detail, name='blog-detail'),
    url(r'blog/overview/$', blog_overview, name='blog-overview'),
]
```

URLconf。 URL `/blog/overview/` `blog_overview`URL`blog-detail`slug='overview'`blog_detail`。

URL `/blog/overview/`blog_overview`blog-detail`

```
urlpatterns = [
    url(r'blog/overview/$', blog_overview, name='blog-overview'),
    url(r'blog/(?P<slug>[\w-]+)/$', blog_detail, name='blog-detail'),
]
```

URLDjango 1.9+

`app_name`URLconfURL

```
# In <myapp>/urls.py
from django.conf.urls import url

from .views import overview

app_name = 'myapp'
urlpatterns = [
    url(r'^$', overview, name='overview'),
]
```

URLconf>`'myapp'`。 URL

```
# In <myproject>/urls.py
from django.conf.urls import include, url

urlpatterns = [
    url(r'^myapp/', include('myapp.urls')),
]
```

URL

```
>>> from django.urls import reverse
>>> reverse('myapp:overview')
'/myapp/overview/'
```

URLconf`namespacenamespace`

```
# In <myproject>/urls.py
urlpatterns = [
    url(r'^myapp/', include('myapp.urls', namespace='mynamespace')),
]
```

URL

```
>>> from django.urls import reverse
>>> reverse('myapp:overview')
'/myapp/overview/'
>>> reverse('mynamespace:overview')
'/myapp/overview/'
```

。

URL https://riptutorial.com/zh-TW/django/topic/3299/url

12:

◦

dictTEMPLATE_CONTEXT_PROCESSORS ◦

Examples

settings.DEBUG

myapp/context_processors.py

```
from django.conf import settings

def debug(request):
  return {'DEBUG': settings.DEBUG}
```

settings.py

```
TEMPLATES = [
    {
        ...
        'OPTIONS': {
            'context_processors': [
                ...
                'myapp.context_processors.debug',
            ],
        },
    },
]
```

<1.9

```
TEMPLATE_CONTEXT_PROCESSORS = (
    ...
    'myapp.context_processors.debug',
)
```

```
 {% if DEBUG %} .header { background:#f00; } {% endif %}
 {{ DEBUG }}
```

models.pyPostdate_published◦

1

app`context_processors.py`

```
from myapp.models import Post
```

```
def recent_blog_posts(request):
    return {'recent_posts':Post.objects.order_by('-date_published')[0:3],}  # Can change
numbers for more/fewer posts
```

2

TEMPLATESsettings.py

```
TEMPLATES = [
    {
        ...
        'OPTIONS': {
            'context_processors': [
                ...
                'myapp.context_processors.recent_blog_posts',
            ],
        },
    },
]
```

1.9DjangoTEMPLATE_CONTEXT_PROCESSORS settings.pysettings.py。

3

recent_blog_posts。

home.html

```
<div class="blog_post_sidebar">
    {% for post in recent_blog_posts %}
        <div class="post">
            <a href="{{post.get_absolute_url}}">{{post.title}}</a>
        </div>
    {% endfor %}
</div>
```

blog.html

```
<div class="content">
    {% for post in recent_blog_posts %}
        <div class="post_detail">
            <h2>{{post.title}}</h2>
            <p>Published on {{post.date_published}}</p>
            <p class="author">Written by: {{post.author}}</p>
            <p><a href="{{post.get_absolute_url}}">Permalink</a></p>
            <p class="post_body">{{post.body}}</p>
        </div>
    {% endfor %}
</div>
```

/。/。

MYAPP / context_processors.py

```
def template_selection(request):
    site_template = 'template_public.html'
    if request.user.is_authenticated():
        if request.user.groups.filter(name="some_group_name").exists():
            site_template = 'template_new.html'

    return {
        'site_template': site_template,
    }
```

。

。

```
{% extends site_template %}
```

https://riptutorial.com/zh-TW/django/topic/491/

13:

Django/Django。

settings.py `MIDDLEWARE_CLASSES`。 Django

```
MIDDLEWARE_CLASSES = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.auth.middleware.SessionAuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]
```

`views.py`1.10`process_response`。 Cross Site Request Forgerycsrf。

。 csrf`CsrfViewMiddleware`。

Examples

Django。 META。

```
class SubdomainMiddleware:
    def process_request(self, request):
        """
        Parse out the subdomain from the request
        """
        host = request.META.get('HTTP_HOST', '')
        host_s = host.replace('www.', '').split('.')
        request.subdomain = None
        if len(host_s) > 2:
            request.subdomain = host_s[0]
```

。 。 DNS。

```
class OrganizationMiddleware:
    def process_request(self, request):
        """
        Determine the organization based on the subdomain
        """
        try:
            request.org = Organization.objects.get(domain=request.subdomain)
        except Organization.DoesNotExist:
            request.org = None
```

。 。

```
MIDDLEWARE_CLASSES = [
```

```
    ...
    'myapp.middleware.SubdomainMiddleware',
    'myapp.middleware.OrganizationMiddleware',
    ...
]
```

IP

```
yourproject/yourapp/middleware
```

settings.pyurlstemplates...

init .py

yourproject/yourapp/middleware.py ◦

◦ IP**filter_ip_middleware.py**

```
#yourproject/yourapp/middleware/filter_ip_middleware.py
from django.core.exceptions import PermissionDenied

class FilterIPMiddleware(object):
    # Check if client IP address is allowed
    def process_request(self, request):
        allowed_ips = ['192.168.1.1', '123.123.123.123', etc...] # Authorized ip's
        ip = request.META.get('REMOTE_ADDR') # Get client IP address
        if ip not in allowed_ips:
            raise PermissionDenied # If user is not allowed raise Error

       # If IP address is allowed we don't do anything
       return None
```

'settings.py'

settings.py`MIDDLEWARE_CLASSES` ◦

```
MIDDLEWARE_CLASSES = (
    'django.middleware.common.CommonMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
     # Above are Django standard middlewares

     # Now we add here our custom middleware
     'yourapp.middleware.filter_ip_middleware.FilterIPMiddleware'
)
```

◦ `ConfictError(detailed_message)` ◦

HTTP 409Confict。 ◦

```
class ConfictErrorHandlingMiddleware:
    def process_exception(self, request, exception):
        if not isinstance(exception, ConflictError):
            return  # Propagate other exceptions, we only handle ConflictError
        context = dict(confict_details=str(exception))
        return TemplateResponse(request, '409.html', context, status=409)
```

Django 1.10

Django 1.10`process_request``process_response`◦

◦ ◦

◦

`HttpResponse` ◦

```
class MyMiddleware:

    def __init__(self, next_layer=None):
        """We allow next_layer to be None because old-style middlewares
        won't accept any argument.
        """
        self.get_response = next_layer

    def process_request(self, request):
        """Let's handle old-style request processing here, as usual."""
        # Do something with request
        # Probably return None
        # Or return an HttpResponse in some cases

    def process_response(self, request, response):
        """Let's handle old-style response processing here, as usual."""
        # Do something with response, possibly using request.
        return response

    def __call__(self, request):
        """Handle new-style middleware here."""
        response = self.process_request(request)
        if response is None:
            # If process_request returned None, we must call the next middleware or
            # the view. Note that here, we are sure that self.get_response is not
            # None because this method is executed only in new-style middlewares.
            response = self.get_response(request)
        response = self.process_response(request, response)
        return response
```

https://riptutorial.com/zh-TW/django/topic/1721/

14: HStoreField - PostgreSQL

- FooModel.objects.filterfield_name__key_name =''

Examples

HStoreField

`HStoreField`

1. `django.contrib.postgres`INSTALLED_APPS
2. `HStoreExtension` `CreateModelAddFieldHStoreExtension`

```
from django.contrib.postgres.operations import HStoreExtension
from django.db import migrations

class FooMigration(migrations.Migration):
    # put your other migration stuff here
    operations = [
        HStoreExtension(),
        ...
    ]
```

HStoreField

`->HStoreField`

`HStoreField`

```
from django.contrib.postgres.fields import HStoreField
from django.db import models

class Catalog(models.model):
    name = models.CharField(max_length=200)
    titles_to_authors = HStoreField()
```

python`create()`

```
Catalog.objects.create(name='Library of Congress', titles_to_authors={
    'Using HStoreField with Django': 'CrazyPython and la communidad',
    'Flabbergeists and thingamajigs': 'La Artista Fooista',
    'Pro Git': 'Scott Chacon and Ben Straub',
})
```

```
Catalog.objects.filter(titles__Pro_Git='Scott Chacon and Ben Straub')
```

`dictfield_name__contains`

```
Catalog.objects.filter(titles__contains={
        'Pro Git': 'Scott Chacon and Ben Straub'})
```

SQL`@>`。

HStoreField - PostgreSQL https://riptutorial.com/zh-TW/django/topic/2670/hstorefield----postgresql

15: RedisDjango -

django-redis-cachedjango-redis。 RedisSESSION_ENGINE SESSION_ENGINE 。 。

settings.py

```
SESSION_ENGINE = "django.contrib.sessions.backends.cache"
```

Examples

django-redis-cache

Redisdjango-redis-cache。

Redis 。

```
$ pip install django-redis-cache
```

settings.pyCACHESDjango 。

```
CACHES = {
    'default': {
        'BACKEND': 'redis_cache.RedisCache',
        'LOCATION': 'localhost:6379',
        'OPTIONS': {
            'DB': 0,
        }
    }
}
```

django-redis

Redisdjango-redis。

Redis 。

```
$ pip install django-redis
```

settings.pyCACHESDjango 。

```
CACHES = {
    'default': {
        'BACKEND': 'django_redis.cache.RedisCache',
        'LOCATION': 'redis://127.0.0.1:6379/1',
        'OPTIONS': {
            'CLIENT_CLASS': 'django_redis.client.DefaultClient',
        }
    }
```

```
}
```

RedisDjango - https://riptutorial.com/zh-TW/django/topic/4085/redisdjango----

16: Django。

DjangoWebORM。 。 Django。

Examples

Django。

djangomain

```
import os, sys

# Setup environ
sys.path.append(os.getcwd())
os.environ.setdefault("DJANGO_SETTINGS_MODULE", "main.settings")

# Setup django
import django
django.setup()

# rest of your imports go here

from main.models import MyModel

# normal python code that makes use of Django models go here

for obj in MyModel.objects.all():
    print obj
```

```
python main/cli.py
```

Django。 https://riptutorial.com/zh-TW/django/topic/5848/django-

17:

/	
UserProfile	UserProfileDjango 。
create_profile	Django User`post_save` create_profile。

。

Django。

。

Django`sender=User`。

UserUserProfile。 `post_saveUser`Django`post_saveUserUserProfile`。

Django。

。

“”。 。 。 。

。 /。 。

Django。

DjangoScoops

- `save()`。
- 。
-

- 。
- 。
- 。
- 。 `save()init()`。 。

Examples

ProndingDjango

```
from django.db import models
from django.contrib.auth.models import User
from django.db.models.signals import post_save
```

```
class UserProfile(models.Model):
    user = models.OneToOneField(User, related_name='user')
    website = models.URLField(default='', blank=True)
    bio = models.TextField(default='', blank=True)

def create_profile(sender, **kwargs):
    user = kwargs["instance"]
    if kwargs["created"]:
        user_profile = UserProfile(user=user)
        user_profile.save()
post_save.connect(create_profile, sender=User)
```

/

```
from django.db import models
from django.contrib.auth.models import User
from django.db.models.signals import post_save
from django.dispatch import receiver

class UserProfile(models.Model):
    user = models.OneToOneField(User, related_name='user')
    website = models.URLField(default='', blank=True)
    bio = models.TextField(default='', blank=True)

@receiver(post_save, sender=UserProfile)
def post_save_user(sender, **kwargs):
    user = kwargs.get('instance')
    if kwargs.get('created'):
        ...
```

pre_save

pre_savesave◦

```
    @receiver(pre_save, sender=User)
    def pre_save_user(sender, instance, **kwargs):
        if not instance._state.adding:
            print ('this is an update')
        else:
            print ('this is an insert')
```

save pre_save

- this is an update ◦
- this is an insert◦

◦

Django◦

```
class Event(models.Model):
    user = models.ForeignKey(User)
```

```
class StatusChange(Event):
    ...


class Comment(Event):
    ...


def send_activity_notification(sender, instance: Event, raw: bool, **kwargs):
    """
    Fire a notification upon saving an event
    """

    if not raw:
        msg_factory = MessageFactory(instance.id)
        msg_factory.on_activity(str(instance))
post_save.connect(send_activity_notification, Event)
```

。

```
post_save.connect(send_activity_notification, StatusChange)
post_save.connect(send_activity_notification, Comment)
```

Python 3.6。

```
class Event(models.Model):

    @classmethod
    def __init_subclass__(cls, **kwargs):
        super().__init_subclass__(**kwargs)
        post_save.connect(send_activity_notification, cls)
```

https://riptutorial.com/zh-TW/django/topic/2555/

18:

Examples

-

Django。td。test_view.py。

```
from django.test import Client, TestCase

class ViewTest(TestCase):

    def test_hello(self):
        c = Client()
        resp = c.get('/hello/')
        self.assertEqual(resp.status_code, 200)
```

```
 ./manage.py test
```

。

```
Traceback (most recent call last):
  File "/home/me/workspace/td/tests_view.py", line 9, in test_hello
    self.assertEqual(resp.status_code, 200)
AssertionError: 200 != 404
```

。 views.py

```
from django.http import HttpResponse
def hello(request):
    return HttpResponse('hello')
```

url py/ hello /

```
from td import views

urlpatterns = [
    url(r'^admin/', include(admin.site.urls)),
    url(r'^hello/', views.hello),
    ....
]
```

`./manage.py test`!!

```
Creating test database for alias 'default'...
.
----------------------------------------------------------------------
Ran 1 test in 0.004s

OK
```

Django

```
from django.db import models

class Author(models.Model):
   name = models.CharField(max_length=50)

    def __str__(self):
        return self.name

    def get_absolute_url(self):
        return reverse('view_author', args=[str(self.id)])


class Book(models.Model):
    author = models.ForeignKey(Manufacturer, on_delete=models.CASCADE)
    private = models.BooleanField(default=false)
    publish_date = models.DateField()

    def get_absolute_url(self):
        return reverse('view_book', args=[str(self.id)])

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

```
from django.test import TestCase
from .models import Book, Author

class BaseModelTestCase(TestCase):

    @classmethod
    def setUpClass(cls):
        super(BaseModelTestCase, cls).setUpClass()
        cls.author = Author(name='hawking')
        cls.author.save()
        cls.first_book = Book(author=cls.author, name="short_history_of_time")
        cls.first_book.save()
        cls.second_book = Book(author=cls.author, name="long_history_of_time")
        cls.second_book.save()


class AuthorModelTestCase(BaseModelTestCase):
    def test_created_properly(self):
         self.assertEqual(self.author.name, 'hawking')
         self.assertEqual(True, self.first_book in self.author.book_set.all())

    def test_absolute_url(self):
        self.assertEqual(self.author.get_absolute_url(), reverse('view_author',
args=[str(self.author.id)]))

class BookModelTestCase(BaseModelTestCase):

    def test_created_properly(self:
        ...
        self.assertEqual(1, len(Book.objects.filter(name__startswith='long'))

    def test_absolute_url(self):
        ...
```

- `created_properly` testsdjango。 file_upload_paths。
- `absolute_url`url
- `mock`
- `BaseModelTestCase`。

。 。

Django

tl; dr `useranother_user` 。 `Client`。

- `self.client` `user`
- `self.another_client` `another_user`
- `self.unlogged_client` `self.unlogged_client`

URL。 `Book`private `public` 。

```
from django.test import TestCase, RequestFactory, Client
from django.core.urlresolvers import reverse

class BaseViewTestCase(TestCase):

    @classmethod
    def setUpClass(cls):
        super(BaseViewTestCase, cls).setUpClass()
        cls.client = Client()
        cls.another_client = Client()
        cls.unlogged_client = Client()
        cls.user = User.objects.create_user(
                'dummy',password='dummy'
                )
        cls.user.save()
        cls.another_user = User.objects.create_user(
                'dummy2', password='dummy2'
                )
        cls.another_user.save()
        cls.first_book = Book.objects.create(
                name='first',
                private = true
        )
        cls.first_book.readers.add(cls.user)
        cls.first_book.save()
        cls.public_book = Template.objects.create(
                name='public',
                private=False
        )
        cls.public_book.save()


    def setUp(self):
        self.client.login(username=self.user.username, password=self.user.username)
        self.another_client.login(username=self.another_user.username,
password=self.another_user.username)


"""
```

```
   Only cls.user owns the first_book and thus only he should be able to see it.
   Others get 403(Forbidden) error
"""
class PrivateBookAccessTestCase(BaseViewTestCase):

    def setUp(self):
        super(PrivateBookAccessTestCase, self).setUp()
        self.url = reverse('view_book',kwargs={'book_id':str(self.first_book.id)})

    def test_user_sees_own_book(self):
        response = self.client.get(self.url)
        self.assertEqual(200, response.status_code)
        self.assertEqual(self.first_book.name,response.context['book'].name)
        self.assertTemplateUsed('myapp/book/view_template.html')

    def test_user_cant_see_others_books(self):
        response = self.another_client.get(self.url)
        self.assertEqual(403, response.status_code)

    def test_unlogged_user_cant_see_private_books(self):
        response = self.unlogged_client.get(self.url)
        self.assertEqual(403, response.status_code)

"""
    Since book is public all three clients should be able to see the book
"""
 class PublicBookAccessTestCase(BaseViewTestCase):

    def setUp(self):
        super(PublicBookAccessTestCase, self).setUp()
        self.url = reverse('view_book',kwargs={'book_id':str(self.public_book.id)})

    def test_user_sees_book(self):
        response = self.client.get(self.url)
        self.assertEqual(200, response.status_code)
        self.assertEqual(self.public_book.name,response.context['book'].name)
        self.assertTemplateUsed('myapp/book/view_template.html')

    def test_another_user_sees_public_books(self):
        response = self.another_client.get(self.url)
        self.assertEqual(200, response.status_code)

    def test_unlogged_user_sees_public_books(self):
        response = self.unlogged_client.get(self.url)
        self.assertEqual(200, response.status_code)
```

Django。。

```
from django.test import TestCase
from myapp.models import Thing

class MyTest(TestCase):

    def test_1(self):
        self.assertEqual(Thing.objects.count(), 0)
        Thing.objects.create()
        self.assertEqual(Thing.objects.count(), 1)

    def test_2(self):
        self.assertEqual(Thing.objects.count(), 0)
```

```
        Thing.objects.create(attr1="value")
        self.assertEqual(Thing.objects.count(), 1)
```

`setUp` djangofixtures

```
class MyTest(TestCase):
    fixtures = ["fixture1.json", "fixture2.json"]
```

django`fixtures`fixtures `FIXTURE_DIRS`

```
# myapp/settings.py
FIXTURE_DIRS = [
    os.path.join(BASE_DIR, 'path', 'to', 'directory'),
]
```

```
# models.py
from django.db import models


class Person(models.Model):
    """A person defined by his/her first- and lastname."""
    firstname = models.CharField(max_length=255)
    lastname = models.CharField(max_length=255)
```

.json

```
# fixture1.json
[
    { "model": "myapp.person",
        "pk": 1,
        "fields": {
            "firstname": "Peter",
            "lastname": "Griffin"
        }
    },
    { "model": "myapp.person",
        "pk": 2,
        "fields": {
            "firstname": "Louis",
            "lastname": "Griffin"
        }
    },
]
```

management-command keepdbshorthand `-k`

```
# Reuse the test-database (since django version 1.8)
$ python manage.py test --keepdb
```

`manage.py testmanage.py test`

```
# Run only tests for the app names "app1"
$ python manage.py test app1
```

```
# If you split the tests file into a module with several tests files for an app
$ python manage.py test app1.tests.test_models

# it's possible to dig down to individual test methods.
$ python manage.py test app1.tests.test_models.MyTestCase.test_something
```

。

```
$ python manage.py test -p test_models*
Creating test database for alias 'default'...
..............................................
----------------------------------------------------------------------
Ran 115 tests in 3.869s

OK
```

--failfast。

```
$ python manage.py test app1
...F..
----------------------------------------------------------------------
Ran 6 tests in 0.977s

FAILED (failures=1)


$ python manage.py test app1 --failfast
...F
======================================================================
[Traceback of the failing test]
----------------------------------------------------------------------
Ran 4 tests in 0.372s

FAILED (failures=1)
```

https://riptutorial.com/zh-TW/django/topic/1232/

19:

- gettext
- ngettext
- ugettext
- ungettext
- pgettext
- npgettextcontextsingularpluralnumber
- gettext_lazy
- ngettext_lazy=
- ugettext_lazy
- ungettext_lazy=
- pgettext_lazycontextmessage
- npgettext_lazycontextsingularpluralnumber = None
- gettext_noop
- ugettext_noop

Examples

settings.py

```
from django.utils.translation import ugettext_lazy as _

USE_I18N = True  # Enable Internationalization
LANGUAGE_CODE = 'en'  # Language in which original texts are written
LANGUAGES = [  # Available languages
    ('en', _("English")),
    ('de', _("German")),
    ('fr', _("French")),
]

# Make sure the LocaleMiddleware is included, AFTER SessionMiddleware
# and BEFORE middlewares using internationalization (such as CommonMiddleware)
MIDDLEWARE_CLASSES = [
   'django.contrib.sessions.middleware.SessionMiddleware',
   'django.middleware.locale.LocaleMiddleware',
   'django.middleware.common.CommonMiddleware',
]
```

◦ gettext ◦

```
from django.utils.translation import ugettext_lazy as _
# It is common to import gettext as the shortcut `_` as it is often used
# several times in the same file.

class Child(models.Model):
```

```
    class Meta:
        verbose_name = _("child")
        verbose_name_plural = _("children")

    first_name = models.CharField(max_length=30, verbose_name=_("first name"))
    last_name = models.CharField(max_length=30, verbose_name=_("last name"))
    age = models.PositiveSmallIntegerField(verbose_name=_("age"))
```

_()◦ ◦

◦ ◦ ◦

◦

```
>>> from django.utils.translation import activate, ugettext as _
>>> month = _("June")
>>> month
'June'
>>> activate('fr')
>>> _("June")
'juin'
>>> activate('de')
>>> _("June")
'Juni'
>>> month
'June'
```

◦

```
>>> from django.utils.translation import activate, ugettext_lazy as _
>>> month = _("June")
>>> month
<django.utils.functional.lazy.<locals>.__proxy__ object at 0x7f61cb805780>
>>> str(month)
'June'
>>> activate('fr')
>>> month
<django.utils.functional.lazy.<locals>.__proxy__ object at 0x7f61cb805780>
>>> "month: {}".format(month)
'month: juin'
>>> "month: %s" % month
'month: Juni'
```

- _("some string")
-

i18n◦

```
{% load i18n %}
```

trans◦

```
{% trans "Some translatable text" %}
{# equivalent to python `ugettext("Some translatable text")` #}
```

trans

```
{% trans "May" context "month" %}
{# equivalent to python `pgettext("May", "month")` #}
```

```
_("My name is {first_name} {last_name}").format(first_name="John", last_name="Doe")
```

blocktrans

```
{% blocktrans with first_name="John" last_name="Doe" %}
  My name is {{ first_name }} {{ last_name }}
{% endblocktrans %}
```

"John""Doe"

```
{% blocktrans with first_name=user.first_name last_name=user.last_name|title %}
  My name is {{ first_name }} {{ last_name }}
{% endblocktrans %}
```

first_namelast_namewith

```
{% blocktrans %}My name is {{ first_name }} {{ last_name }}{% endblocktrans %}
```

“”。

```
{% blocktrans %}
    My name is {{ user.first_name }} {{ user.last_name }}
{% endblocktrans %}
```

。

blocktrans。

```
{% blocktrans count nb=users|length }}
    There is {{ nb }} user.
{% plural %}
    There are {{ nb }} users.
{% endblocktrans %}
```

i18n_("")。

```
{{ site_name|default:_("It works!") }}
{% firstof var1 var2 _("translatable fallback") %}
```

django。 _("It works!")default'_("It works!")'name"name"。

。 djangomakemessages 。

```
$ django-admin makemessages -l fr
processing locale fr
```

◦ app myapp myapp/locale/fr/LC_MESSAGES/django.po ◦

```
# SOME DESCRIPTIVE TITLE
# Copyright (C) YEAR THE PACKAGE'S COPYRIGHT HOLDER
# This file is distributed under the same license as the PACKAGE package.
# FIRST AUTHOR <EMAIL@ADDRESS>, YEAR
#
#, fuzzy
msgid ""
msgstr ""
"Project-Id-Version: PACKAGE VERSION\n"
"Report-Msgid-Bugs-To: \n"
"POT-Creation-Date: 2016-07-24 14:01+0200\n"
"PO-Revision-Date: YEAR-MO-DA HO:MI+ZONE\n"
"Last-Translator: FULL NAME <EMAIL@ADDRESS>\n"
"Language-Team: LANGUAGE <LL@li.org>\n"
"Language: \n"
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=UTF-8\n"
"Content-Transfer-Encoding: 8bit\n"

#: myapp/models.py:22
msgid "user"
msgstr ""

#: myapp/models.py:39
msgid "A user already exists with this email address."
msgstr ""

#: myapp/templates/myapp/register.html:155
#, python-format
msgid ""
"By signing up, you accept our <a href=\"%(terms_url)s\" "
"target=_blank>Terms of services</a>."
msgstr ""
```

◦ ◦ msgid◦ msgstr◦

◦

```
#: myapp/templates/myapp/register.html:155
#, python-format
msgid ""
"By signing up, you accept our <a href=\"%(terms_url)s\" "
"target=_blank>Terms of services</a>."
msgstr ""
"En vous inscrivant, vous acceptez nos <a href=\"%(terms_url)s\" "
"target=_blank>Conditions d'utilisation</a>"
```

.po.mo◦ compilemessages

```
$ django-admin compilemessages
```

◦

django-admin makemessages -l fr ◦ .po◦ ◦ .podjango-admin makemessages -a ◦ .podjango-admin compilemessages.mo◦

Noop

(u)gettext_noop◦

◦ **gettext** ◦ ◦

```
# THIS WILL NOT WORK AS EXPECTED
import logging
from django.contrib import messages

logger = logging.getLogger(__name__)

error_message = "Oops, something went wrong!"
logger.error(error_message)
messages.error(request, _(error_message))
```

.po◦ gettext_noop ◦

```
error_message = ugettext_noop("Oops, something went wrong!")
logger.error(error_message)
messages.error(request, _(error_message))
```

"Oops, something went wrong!".po◦ ◦

makemessages◦ .pofuzzy

```
#: templates/randa/map.html:91
#, fuzzy
msgid "Country"
msgstr "Länderinfo"
```

fuzzy◦

makemessagespython◦

```
translation = _("firstline"
"secondline"
"thirdline")
```

firstline◦ ◦

https://riptutorial.com/zh-TW/django/topic/2579/

20:

CBV。 django。

Classy Class Based View。

Examples

。

。 TemplateView 。 。 。

views.py

```
from django.views.generic import TemplateView


class AboutView(TemplateView):
    template_name = "about.html"
```

urls.py

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

urlpatterns = [
    url('^about/', views.AboutView.as_view(), name='about'),
]
```

urlAboutView。 as_view()。

。 。 get_context_data。

views.py

```
class BookView(DetailView):
    template_name = "book.html"

    def get_context_data(self, **kwargs)
        """ get_context_data let you fill the template context """
        context = super(BookView, self).get_context_data(**kwargs)
        # Get Related publishers
        context['publishers'] = self.object.publishers.filter(is_active=True)
        return context
```

get_context_data。。

book.html

```
<h3>Active publishers</h3>
<ul>
  {% for publisher in publishers %}
    <li>{{ publisher.name }}</li>
  {% endfor %}
</ul>
```

get_context_data。

ListViewDetailView

/ models.py

```
from django.db import models

class Pokemon(models.Model):
    name = models.CharField(max_length=24)
    species = models.CharField(max_length=48)
    slug = models.CharField(max_length=48)
```

/ views.py

```
from django.views.generic import ListView, DetailView
from .models import Pokemon


class PokedexView(ListView):
    """ Provide a list of Pokemon objects """
    model = Pokemon
    paginate_by = 25

class PokemonView(DetailView):
    model = Pokemon
```

。。template_name。。

/// pokemon_list.html

```
<!DOCTYPE html>
<title>Pokedex</title>
<ul>{% for pokemon in pokemon_list %}
    <li><a href="{% url "app:pokemon" pokemon.pk %}">{{ pokemon.name }}</a>
```

```
        &ndash; {{ pokemon.species }}
</ul>
```

object_listpokemon_list ◦ ◦ Paginator◦

/// pokemon_detail.html

```
<!DOCTYPE html>
<title>Pokemon {{ pokemon.name }}</title>
<h1>{{ pokemon.name }}</h1>
<h2>{{ pokemon.species }} </h2>
```

objectpokemon◦

/ urls.py

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

app_name = 'app'
urlpatterns = [
    url(r'^pokemon/$', views.PokedexView.as_view(), name='pokedex'),
    url(r'^pokemon/(?P<pk>\d+)/$', views.PokemonView.as_view(), name='pokemon'),
]
```

◦ slug◦ ◦ slug◦

```
url(r'^pokemon/(?P<slug>[A-Za-z0-9_-]+)/$', views.PokemonView.as_view(), name='pokemon'),
```

slugDetailViewslug_field◦

URL◦

◦ ◦ ◦

/ views.py

```
from django.core.urlresolvers import reverse_lazy
from django.views.generic.edit import CreateView, UpdateView, DeleteView
from .models import Pokemon


class PokemonCreate(CreateView):
    model = Pokemon
    fields = ['name', 'species']
```

```
class PokemonUpdate(UpdateView):
    model = Pokemon
    fields = ['name', 'species']


class PokemonDelete(DeleteView):
    model = Pokemon
    success_url = reverse_lazy('pokedex')
```

CreateViewUpdateView modelfields。 “_form”。 template_name_suffix。 DeleteView。

UpdateViewDeleteView。 DetailViewurl。

app / templates / app / pokemon_form.html

```
<form action="" method="post">
    {% csrf_token %}
    {{ form.as_p }}
    <input type="submit" value="Save" />
</form>
```

form。 as_p as_p 。

app / templates / app / pokemon_confirm_delete.html

```
<form action="" method="post">
    {% csrf_token %}
    <p>Are you sure you want to delete "{{ object }}"?</p>
    <input type="submit" value="Confirm" />
</form>
```

djangocsrf_token。 url/url。

。 URL。 pokemonget_absolute_url。 。 。

/ models.py

```
from django.db import models
from django.urls import reverse
from django.utils.encoding import python_2_unicode_compatible


@python_2_unicode_compatible
class Pokemon(models.Model):
    name = models.CharField(max_length=24)
    species = models.CharField(max_length=48)
```

```
    def get_absolute_url(self):
        return reverse('app:pokemon', kwargs={'pk':self.pk})

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

python 2。

views.py

```
from django.http import HttpResponse
from django.views.generic import View

class MyView(View):
    def get(self, request):
        # <view logic>
        return HttpResponse('result')
```

urls.py

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

urlpatterns = [
    url(r'^about/$', MyView.as_view()),
]
```

Django»

DjangoCreateView

CRUD。DjangoCRUD。CBV。

`CreateView`3 - URL。

```
from django.views.generic import CreateView
from .models import Campaign

class CampaignCreateView(CreateView):
    model = Campaign
    fields = ('title', 'description')

    success_url = "/campaigns/list"
```

`success_url` 。 `get_success_urlreversereverse_lazy`url。

。 `<app name>/<model name>_form.html`。 。 dashboard dashboard/campaign_form.html。

`form`。

```
<form action="" method="post">
```

```
    {% csrf_token %}
    {{ form.as_p }}
    <input type="submit" value="Save" />
</form>
```

。

```
url('^campaign/new/$', CampaignCreateView.as_view(), name='campaign_new'),
```

。。URL。。

Django。

```
from django.contrib import messages
from django.views.generic import TemplateView

from .forms import AddPostForm, AddCommentForm
from .models import Comment

class AddCommentView(TemplateView):

    post_form_class = AddPostForm
    comment_form_class = AddCommentForm
    template_name = 'blog/post.html'

    def post(self, request):
        post_data = request.POST or None
        post_form = self.post_form_class(post_data, prefix='post')
        comment_form = self.comment_form_class(post_data, prefix='comment')

        context = self.get_context_data(post_form=post_form,
                                        comment_form=comment_form)

        if post_form.is_valid():
            self.form_save(post_form)
        if comment_form.is_valid():
            self.form_save(comment_form)

        return self.render_to_response(context)

    def form_save(self, form):
        obj = form.save()
        messages.success(self.request, "{} saved successfully".format(obj))
        return obj

    def get(self, request, *args, **kwargs):
        return self.post(request, *args, **kwargs)
```

https://riptutorial.com/zh-TW/django/topic/1220/

21:

Examples

```
class Skill(models.Model):
    name = models.CharField(max_length=50)
    description = models.TextField()

class Developer(models.Model):
    name = models.CharField(max_length=50)
    skills = models.ManyToManyField(Skill, through='DeveloperSkill')

class DeveloperSkill(models.Model):
    """Developer skills with respective ability and experience."""

    class Meta:
        order_with_respect_to = 'developer'
        """Sort skills per developer so that he can choose which
        skills to display on top for instance.
        """
        unique_together = [
            ('developer', 'skill'),
        ]
        """It's recommended that a together unique index be created on
        `(developer,skill)`. This is especially useful if your database is
        being access/modified from outside django. You will find that such an
        index is created by django when an explicit through model is not
        being used.
        """


    ABILITY_CHOICES = [
        (1, "Beginner"),
        (2, "Accustomed"),
        (3, "Intermediate"),
        (4, "Strong knowledge"),
        (5, "Expert"),
    ]

    developer = models.ForeignKey(Developer, models.CASCADE)
    skill = models.ForeignKey(Skill, models.CASCADE)
    """The many-to-many relation between both models is made by the
    above two foreign keys.

    Other fields (below) store information about the relation itself.
    """

    ability = models.PositiveSmallIntegerField(choices=ABILITY_CHOICES)
    experience = models.PositiveSmallIntegerField(help_text="Years of experience.")
```

`(developer,skill)`。 django/。 django。

。

```
class Person(models.Model):
```

```
    name = models.CharField(max_length=50)
    description = models.TextField()

class Club(models.Model):
    name = models.CharField(max_length=50)
    members = models.ManyToManyField(Person)
```

Person SClub。

django。 myapp_person myapp_clubmyapp_club_members。 Django myapp_club_members(club_id,person_id)。

ManyToMany

```
class Person(models.Model):
    name = models.CharField(max_length=50)
    description = models.TextField()

class Club(models.Model):
    name = models.CharField(max_length=50)
    members = models.ManyToManyField(Person)
```

TomBill

```
tom = Person.objects.create(name="Tom", description="A nice guy")
bill = Person.objects.create(name="Bill", description="Good dancer")

nightclub = Club.objects.create(name="The Saturday Night Club")
nightclub.members.add(tom, bill)
```

```
for person in nightclub.members.all():
    print(person.name)
```

```
Tom
Bill
```

https://riptutorial.com/zh-TW/django/topic/2379/

22: CookiecutterDjango

Examples

Cookiecutterdjango

Cookiecutter

-
- virtualenv
- PostgreSQL

virtualenv

```
$ mkvirtualenv <virtualenv name>
$ workon <virtualenv name>
```

Cookiecutter

```
$ pip install cookiecutter
```

◦ django

```
$ cookiecutter https://github.com/pydanny/cookiecutter-django.git
```

cookiecutter-django repocookiecutter。“”[]。

```
project_name [project_name]: example_project
repo_name [example_project]:
author_name [Your Name]: Atul Mishra
email [Your email]: abc@gmail.com
description [A short description of the project.]: Demo Project
domain_name [example.com]: example.com
version [0.1.0]: 0.1.0
timezone [UTC]: UTC
now [2016/03/08]: 2016/03/08
year [2016]: 2016
use_whitenoise [y]: y
use_celery [n]: n
use_mailhog [n]: n
use_sentry [n]: n
use_newrelic [n]: n
use_opbeat [n]: n
windows [n]: n
use_python2 [n]: n
```

◦ ◦

CookiecutterDjango https://riptutorial.com/zh-TW/django/topic/5385/cookiecutterdjango-

23: django

Django。

Examples

Django

/SQLite。 MySQL / Postgres。

app“init.py”。

```
/your_django_project/your_app/migrations
```

django。

```
python manage.py makemigrations
python manage.py migrate
```

django https://riptutorial.com/zh-TW/django/topic/9513/django

24:

Examples

XSS

XSSHTMLJS。 。

Django。

```
context = {
    'class_name': 'large" style="font-size:4000px',
    'paragraph': (
        "<script type=\"text/javascript\">alert('hello world!');</script>"),
}
```

```
<p class="{{ class_name }}">{{ paragraph }}</p>
<!-- Will be rendered as: -->
<p class="large&amp;quot; style=&amp;quot;font-size: 4000px">&amp;lt;script&amp;gt;alert(&amp;#39;hello
world!&amp;#39;);&amp;lt;/script&amp;gt;</p>
```

HTML

```
<p class="{{ class_name|safe }}">{{ paragraph }}</p>
<!-- Will be rendered as: -->
<p class="large" style="font-size: 4000px">&amp;lt;script&amp;gt;alert(&amp;#39;hello
world!&amp;#39;);&amp;lt;/script&amp;gt;</p>
```

```
{% autoescape off %}
<p class="{{ class_name }}">{{ paragraph }}</p>
{% endautoescape %}
<!-- Will be rendered as: -->
<p class="large" style="font-size: 4000px"><script>alert('hello world!');</script></p>
```

```
from django.utils.safestring import mark_safe

context = {
    'class_name': 'large" style="font-size:4000px',
    'paragraph': mark_safe(
        "<script type=\"text/javascript\">alert('hello world!');</script>"),
}
```

```
<p class="{{ class_name }}">{{ paragraph }}</p>
<!-- Will be rendered as: -->
<p class="large&amp;quot; style=&amp;quot;font-size: 4000px"><script>alert('hello
world!');</script></p>
```

Django`format_html`

```
from django.utils.html import format_html
```

```
context = {
    'var': format_html('<b>{}</b> {}', 'hello', '<i>world!</i>'),
}
```

```
<p>{{ var }}</p>
<!-- Will be rendered as -->
<p><b>hello</b> &lt;i&gt;world!&lt;/i&gt;</p>
```

。

XFrameOptionsMiddlewareXFrameOptionsMiddleware。 。

```
# settings.py
MIDDLEWARE_CLASSES = [
    ...
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
    ...
]
```

“X-Frame-Options”。“SAMEORIGIN”。X_FRAME_OPTIONS

```
X_FRAME_OPTIONS = 'DENY'
```

。

```
from django.utils.decorators import method_decorator
from django.views.decorators.clickjacking import (
    xframe_options_exempt, xframe_options_deny, xframe_options_sameorigin,
)

xframe_options_exempt_m = method_decorator(xframe_options_exempt, name='dispatch')

@xframe_options_sameorigin
def my_view(request, *args, **kwargs):
    """Forces 'X-Frame-Options: SAMEORIGIN'."""
    return HttpResponse(...)

@method_decorator(xframe_options_deny, name='dispatch')
class MyView(View):
    """Forces 'X-Frame-Options: DENY'."""

@xframe_options_exempt_m
class MyView(View):
    """Does not set 'X-Frame-Options' header when passing through the
    XFrameOptionsMiddleware.
    """
```

CSRF

CSRFXSRF。

CSRFCsrfViewMiddleware。 。

```
# settings.py
MIDDLEWARE_CLASSES = [
    ...
    'django.middleware.csrf.CsrfViewMiddleware',
    ...
]
```

cookie。 GET HEAD OPTIONSTRACEcookiecsrfmiddlewaretokenX-CsrfToken。 cookie。

HTTPS。 HTTP_REFERERCSRF_TRUSTED_ORIGINS 1.9。

POSTCSRF。 {% csrf_token %}cookie

```
<form method='POST'>
{% csrf_token %}
...
</form>
```

@csrf_exemptCSRF

```
from django.views.decorators.csrf import csrf_exempt

@csrf_exempt
def my_view(request, *args, **kwargs):
    """Allows unsafe methods without CSRF protection"""
    return HttpResponse(...)
```

CSRFCsrfViewMiddleware 。 @csrf_protect

```
from django.views.decorators.csrf import csrf_protect

@csrf_protect
def my_view(request, *args, **kwargs):
    """This view is protected against CSRF attacks if the middleware is disabled"""
    return HttpResponse(...)
```

https://riptutorial.com/zh-TW/django/topic/2957/

25:

Examples

ModelForm

ModelFormModel`ModelForm`

```
from django import forms

class OrderForm(forms.ModelForm):
    class Meta:
        model = Order
        fields = ['item', 'order_date', 'customer', 'status']
```

Django

`django.forms.Form`◦

`CharField URLField IntegerField`◦

```
from django import forms

class ContactForm(forms.Form):
    contact_name = forms.CharField(
        label="Your name", required=True,
        widget=forms.TextInput(attrs={'class': 'form-control'}))
    contact_email = forms.EmailField(
        label="Your Email Address", required=True,
        widget=forms.TextInput(attrs={'class': 'form-control'}))
    content = forms.CharField(
        label="Your Message", required=True,
        widget=forms.Textarea(attrs={'class': 'form-control'}))
```

WidgetDjangoHTMLhtml

`attrs`html◦

`content.render("name", "Your Name")`

```
<input title="Your name" type="text" name="name" value="Your Name" class="form-control" />
```

views.pymodelForm

```
from django.db import models
from django.contrib.auth.models import User

class UserModuleProfile(models.Model):
    user = models.OneToOneField(User)
    expired = models.DateTimeField()
    admin = models.BooleanField(default=False)
```

```
    employee_id = models.CharField(max_length=50)
    organisation_name = models.ForeignKey('Organizations', on_delete=models.PROTECT)
    country = models.CharField(max_length=100)
    position = models.CharField(max_length=100)

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

```
from .models import UserModuleProfile, from django.contrib.auth.models import User
from django import forms

class UserProfileForm(forms.ModelForm):
    admin = forms.BooleanField(label="Make this User
Admin",widget=forms.CheckboxInput(),required=False)
    employee_id = forms.CharField(label="Employee Id ")
    organisation_name = forms.ModelChoiceField(label='Organisation
Name',required=True,queryset=Organizations.objects.all(),empty_label="Select an Organization")
    country = forms.CharField(label="Country")
    position = forms.CharField(label="Position")

    class Meta:
        model = UserModuleProfile
        fields = ('admin','employee_id','organisation_name','country','position',)

    def __init__(self, *args, **kwargs):
        admin_check = kwargs.pop('admin_check', False)
        super(UserProfileForm, self).__init__(*args, **kwargs)
        if not admin_check:
            del self.fields['admin']
```

Meta**init**views.py。。

Meta。

admin。

```
def edit_profile(request,user_id):
    context = RequestContext(request)
    user = get_object_or_404(User, id=user_id)
    profile = get_object_or_404(UserModuleProfile, user_id=user_id)
    admin_check = False
    if request.user.is_superuser:
        admin_check = True
    # If it's a HTTP POST, we're interested in processing form data.
    if request.method == 'POST':
        # Attempt to grab information from the raw form information.
        profile_form =
UserProfileForm(data=request.POST,instance=profile,admin_check=admin_check)
        # If the form is valid...
        if profile_form.is_valid():
            form_bool = request.POST.get("admin", "xxx")
            if form_bool == "xxx":
                form_bool_value = False
            else:
                form_bool_value = True
            profile = profile_form.save(commit=False)
            profile.user = user
            profile.admin = form_bool_value
```

```
            profile.save()
            edited = True
        else:
            print profile_form.errors

    # Not a HTTP POST, so we render our form using ModelForm instance.
    # These forms will be blank, ready for user input.
    else:
        profile_form = UserProfileForm(instance = profile,admin_check=admin_check)

    return render_to_response(
            'usermodule/edit_user.html',
            {'id':user_id, 'profile_form': profile_form, 'edited': edited, 'user':user},
            context)
```

◦ TrueFalse admin_check◦

```
profile_form = UserProfileForm(instance = profile,admin_check=admin_check)
```

initadmin_check◦ False admin◦ adminFalse ◦

```
form_bool = request.POST.get("admin", "xxx")
if form_bool == "xxx":
    form_bool_value = False
else:
    form_bool_value = True
```

Django Forms

MEDIA_ROOTMEDIA_URLsettings.py

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

ImageField Pillow pip install pillow ◦

```
ImportError: No module named PIL
```

PillowPILPython◦ PIL◦

Django FileFieldImageField

forms.py

```
from django import forms


class UploadDocumentForm(forms.Form):
    file = forms.FileField()
    image = forms.ImageField()
```

views.py

```
from django.shortcuts import render
from .forms import UploadDocumentForm


def upload_doc(request):
    form = UploadDocumentForm()
    if request.method == 'POST':
        form = UploadDocumentForm(request.POST, request.FILES)  # Do not forget to add:
request.FILES
        if form.is_valid():
            # Do something with our files or simply save them
            # if saved, our files would be located in media/ folder under the project's base
folder
            form.save()
    return render(request, 'upload_doc.html', locals())
```

upload_doc.html

```
<html>
    <head>File Uploads</head>
    <body>
        <form enctype="multipart/form-data" action="" method="post"> <!-- Do not forget to
add: enctype="multipart/form-data" -->
            {% csrf_token %}
            {{ form }}
            <input type="submit" value="Save">
        </form>
    </body>
</html>
```

DjangoSetPasswordForm 。

。

- -；
- -。

。 __init__()。

```
class EmailChangeForm(forms.Form):
    """
    A form that lets a user change set their email while checking for a change in the
    e-mail.
    """
    error_messages = {
        'email_mismatch': _("The two email addresses fields didn't match."),
        'not_changed': _("The email address is the same as the one already defined."),
    }

    new_email1 = forms.EmailField(
        label=_("New email address"),
        widget=forms.EmailInput,
    )

    new_email2 = forms.EmailField(
        label=_("New email address confirmation"),
        widget=forms.EmailInput,
```

```
    )

    def __init__(self, user, *args, **kwargs):
        self.user = user
        super(EmailChangeForm, self).__init__(*args, **kwargs)

    def clean_new_email1(self):
        old_email = self.user.email
        new_email1 = self.cleaned_data.get('new_email1')
        if new_email1 and old_email:
            if new_email1 == old_email:
                raise forms.ValidationError(
                    self.error_messages['not_changed'],
                    code='not_changed',
                )
        return new_email1

    def clean_new_email2(self):
        new_email1 = self.cleaned_data.get('new_email1')
        new_email2 = self.cleaned_data.get('new_email2')
        if new_email1 and new_email2:
            if new_email1 != new_email2:
                raise forms.ValidationError(
                    self.error_messages['email_mismatch'],
                    code='email_mismatch',
                )
        return new_email2

    def save(self, commit=True):
        email = self.cleaned_data["new_email1"]
        self.user.email = email
        if commit:
            self.user.save()
        return self.user



def email_change(request):
    form = EmailChangeForm()
    if request.method=='POST':
        form = Email_Change_Form(user,request.POST)
        if form.is_valid():
            if request.user.is_authenticated:
                if form.cleaned_data['email1']  == form.cleaned_data['email2']:
                    user = request.user
                    u = User.objects.get(username=user)
                    # get the proper user
                    u.email = form.cleaned_data['email1']
                    u.save()
                    return HttpResponseRedirect("/accounts/profile/")
    else:
        return render_to_response("email_change.html", {'form':form},
                                  context_instance=RequestContext(request))
```

https://riptutorial.com/zh-TW/django/topic/1217/

26:

Examples

◦

models.py

```
from __future__ import unicode_literals
from django.db import models
from django.contrib.auth.models import (
        AbstractBaseUser, BaseUserManager, PermissionsMixin)
from django.utils import timezone
from django.utils.translation import ugettext_lazy as _


class UserManager(BaseUserManager):
    def _create_user(self, email,password, is_staff, is_superuser, **extra_fields):
        now = timezone.now()
        if not email:
            raise ValueError('users must have an email address')
        email = self.normalize_email(email)
        user = self.model(email = email,
                          is_staff = is_staff,
                          is_superuser = is_superuser,
                          last_login = now,
                          date_joined = now,
                          **extra_fields)
        user.set_password(password)
        user.save(using = self._db)
        return user

    def create_user(self, email, password=None, **extra_fields):
        user = self._create_user(email, password, False, False, **extra_fields)
        return user

    def create_superuser(self, email, password, **extra_fields):
        user = self._create_user(email, password, True, True, **extra_fields)
        return user

class User(AbstractBaseUser,PermissionsMixin):
    """My own custom user class"""

    email = models.EmailField(max_length=255, unique=True, db_index=True,
verbose_name=_('email address'))
    date_joined = models.DateTimeField(auto_now_add=True)
    is_active = models.BooleanField(default=True)
    is_staff = models.BooleanField(default=False)

    objects = UserManager()

    USERNAME_FIELD = 'email'
    REQUIRED_FIELDS = []

    class Meta:
        verbose_name = _('user')
```

```
        verbose_name_plural = _('users')

    def get_full_name(self):
    """Return the email."""
        return self.email

    def get_short_name(self):
    """Return the email."""
        return self.email
```

forms.py

```
from django import forms
from django.contrib.auth.forms import UserCreationForm
from .models import User


class RegistrationForm(UserCreationForm):
    email = forms.EmailField(widget=forms.TextInput(
        attrs={'class': 'form-control','type':'text','name': 'email'}),
        label="Email")
    password1 = forms.CharField(widget=forms.PasswordInput(
        attrs={'class':'form-control','type':'password', 'name':'password1'}),
        label="Password")
    password2 = forms.CharField(widget=forms.PasswordInput(
        attrs={'class':'form-control','type':'password', 'name': 'password2'}),
        label="Password (again)")

    '''added attributes so as to customise for styling, like bootstrap'''
    class Meta:
        model = User
        fields = ['email','password1','password2']
        field_order = ['email','password1','password2']

    def clean(self):
    """
    Verifies that the values entered into the password fields match
    NOTE : errors here will appear in 'non_field_errors()'
    """
        cleaned_data = super(RegistrationForm, self).clean()
        if 'password1' in self.cleaned_data and 'password2' in self.cleaned_data:
            if self.cleaned_data['password1'] != self.cleaned_data['password2']:
                raise forms.ValidationError("Passwords don't match. Please try again!")
        return self.cleaned_data

    def save(self, commit=True):
        user = super(RegistrationForm,self).save(commit=False)
        user.set_password(self.cleaned_data['password1'])
        if commit:
            user.save()
        return user

#The save(commit=False) tells Django to save the new record, but dont commit it to the
database yet

class AuthenticationForm(forms.Form): # Note: forms.Form NOT forms.ModelForm
    email = forms.EmailField(widget=forms.TextInput(
        attrs={'class': 'form-control','type':'text','name': 'email','placeholder':'Email'}),
        label='Email')
    password = forms.CharField(widget=forms.PasswordInput(
```

```
        attrs={'class':'form-control','type':'password', 'name':
'password','placeholder':'Password'}),
        label='Password')

    class Meta:
        fields = ['email', 'password']
```

views.py

```
from django.shortcuts import redirect, render, HttpResponse
from django.contrib.auth import login as django_login, logout as django_logout, authenticate
as django_authenticate
#importing as such so that it doesn't create a confusion with our methods and django's default
methods

from django.contrib.auth.decorators import login_required
from .forms import AuthenticationForm, RegistrationForm


def login(request):
    if request.method == 'POST':
        form = AuthenticationForm(data = request.POST)
        if form.is_valid():
            email = request.POST['email']
            password = request.POST['password']
            user = django_authenticate(email=email, password=password)
            if user is not None:
                if user.is_active:
                    django_login(request,user)
                    return redirect('/dashboard') #user is redirected to dashboard
    else:
        form = AuthenticationForm()

    return render(request,'login.html',{'form':form,})

def register(request):
    if request.method == 'POST':
        form = RegistrationForm(data = request.POST)
        if form.is_valid():
            user = form.save()
            u = django_authenticate(user.email = user, user.password = password)
            django_login(request,u)
            return redirect('/dashboard')
    else:
        form = RegistrationForm()

    return render(request,'register.html',{'form':form,})

def logout(request):
    django_logout(request)
    return redirect('/')

@login_required(login_url ="/")
def dashboard(request):
    return render(request, 'dashboard.html',{})
```

settings.py

```
AUTH_USER_MODEL = 'myapp.User'
```

admin.py

```
from django.contrib import admin
from django.contrib.auth.admin import UserAdmin as BaseUserAdmin
from django.contrib.auth.models import Group
from .models import User


class UserAdmin(BaseUserAdmin):
    list_display = ('email','is_staff')
    list_filter = ('is_staff',)
    fieldsets = ((None,
                  {'fields':('email','password')}), ('Permissions',{'fields':('is_staff',)}),)
    add_fieldsets = ((None, {'classes': ('wide',), 'fields': ('email', 'password1',
'password2')}),)
    search_fields =('email',)
    ordering = ('email',)
    filter_horizontal = ()

admin.site.register(User, UserAdmin)
admin.site.unregister(Group)
```

`email``username`

usernameemailAbstractBaseUserAbstractUserUser◦ usernameemailAbstractUser◦ AbstractUser◦

```
from django.contrib.auth.models import (
    AbstractBaseUser, PermissionsMixin, BaseUserManager,
)
from django.db import models
from django.utils import timezone
from django.utils.translation import ugettext_lazy as _

class UserManager(BaseUserManager):

    use_in_migrations = True

    def _create_user(self, email, password, **extra_fields):
        if not email:
            raise ValueError('The given email must be set')
        email = self.normalize_email(email)
        user = self.model(email=email, **extra_fields)
        user.set_password(password)
        user.save(using=self._db)
        return user

    def create_user(self, email, password=None, **extra_fields):
        extra_fields.setdefault('is_staff', False)
        extra_fields.setdefault('is_superuser', False)
        return self._create_user(email, password, **extra_fields)

    def create_superuser(self, email, password, **extra_fields):
        extra_fields.setdefault('is_staff', True)
        extra_fields.setdefault('is_superuser', True)

        if extra_fields.get('is_staff') is not True:
            raise ValueError('Superuser must have is_staff=True.')
        if extra_fields.get('is_superuser') is not True:
```

```
            raise ValueError('Superuser must have is_superuser=True.')

    return self._create_user(email, password, **extra_fields)


class User(AbstractBaseUser, PermissionsMixin):
    """PermissionsMixin contains the following fields:
        - `is_superuser`
        - `groups`
        - `user_permissions`
     You can omit this mix-in if you don't want to use permissions or
     if you want to implement your own permissions logic.
     """

    class Meta:
        verbose_name = _("user")
        verbose_name_plural = _("users")
        db_table = 'auth_user'
        # `db_table` is only needed if you move from the existing default
        # User model to a custom one. This enables to keep the existing data.

    USERNAME_FIELD = 'email'
    """Use the email as unique username."""

    REQUIRED_FIELDS = ['first_name', 'last_name']

    GENDER_MALE = 'M'
    GENDER_FEMALE = 'F'
    GENDER_CHOICES = [
        (GENDER_MALE, _("Male")),
        (GENDER_FEMALE, _("Female")),
    ]

    email = models.EmailField(
        verbose_name=_("email address"), unique=True,
        error_messages={
            'unique': _(
                "A user is already registered with this email address"),
        },
    )
    gender = models.CharField(
        max_length=1, blank=True, choices=GENDER_CHOICES,
        verbose_name=_("gender"),
    )
    first_name = models.CharField(
        max_length=30, verbose_name=_("first name"),
    )
    last_name = models.CharField(
        max_length=30, verbose_name=_("last name"),
    )
    is_staff = models.BooleanField(
        verbose_name=_("staff status"),
        default=False,
        help_text=_(
            "Designates whether the user can log into this admin site."
        ),
    )
    is_active = models.BooleanField(
        verbose_name=_("active"),
        default=True,
        help_text=_(
```

```
            "Designates whether this user should be treated as active. "
            "Unselect this instead of deleting accounts."
        ),
    )
    date_joined = models.DateTimeField(
        verbose_name=_("date joined"), default=timezone.now,
    )

    objects = UserManager()
```

Django

UserProfile

OneToOneUserUserProfile

```
from django.db import models
from django.contrib.auth.models import User
from django.db.models.signals import post_save

class UserProfile(models.Model):
    user = models.OneToOneField(User, related_name='user')
    photo = FileField(verbose_name=_("Profile Picture"),
                      upload_to=upload_to("main.UserProfile.photo", "profiles"),
                      format="Image", max_length=255, null=True, blank=True)
    website = models.URLField(default='', blank=True)
    bio = models.TextField(default='', blank=True)
    phone = models.CharField(max_length=20, blank=True, default='')
    city = models.CharField(max_length=100, default='', blank=True)
    country = models.CharField(max_length=100, default='', blank=True)
    organization = models.CharField(max_length=100, default='', blank=True)
```

Django

Django SignalsUserProfile User◦ UserProfile◦ ◦

```
def create_profile(sender, **kwargs):
    user = kwargs["instance"]
    if kwargs["created"]:
        user_profile = UserProfile(user=user)
        user_profile.save()
post_save.connect(create_profile, sender=User)
```

inlineformset_factory

views.py

```
from django.shortcuts import render, HttpResponseRedirect
from django.contrib.auth.decorators import login_required
from django.contrib.auth.models import User
from .models import UserProfile
from .forms import UserForm
from django.forms.models import inlineformset_factory
from django.core.exceptions import PermissionDenied
@login_required() # only logged in users should access this
def edit_user(request, pk):
```

```
    # querying the User object with pk from url
    user = User.objects.get(pk=pk)

    # prepopulate UserProfileForm with retrieved user values from above.
    user_form = UserForm(instance=user)

    # The sorcery begins from here, see explanation https://blog.khophi.co/extending-django-
user-model-userprofile-like-a-pro/
    ProfileInlineFormset = inlineformset_factory(User, UserProfile, fields=('website', 'bio',
'phone', 'city', 'country', 'organization'))
    formset = ProfileInlineFormset(instance=user)

    if request.user.is_authenticated() and request.user.id == user.id:
        if request.method == "POST":
            user_form = UserForm(request.POST, request.FILES, instance=user)
            formset = ProfileInlineFormset(request.POST, request.FILES, instance=user)

            if user_form.is_valid():
                created_user = user_form.save(commit=False)
                formset = ProfileInlineFormset(request.POST, request.FILES,
instance=created_user)

                if formset.is_valid():
                    created_user.save()
                    formset.save()
                    return HttpResponseRedirect('/accounts/profile/')

        return render(request, "account/account_update.html", {
            "noodle": pk,
            "noodle_form": user_form,
            "formset": formset,
        })
    else:
        raise PermissionDenied
```

account_update.html

```
{% load material_form %}
<!-- Material form is just a materialize thing for django forms -->
<div class="col s12 m8 offset-m2">
      <div class="card">
        <div class="card-content">
        <h2 class="flow-text">Update your information</h2>
          <form action="." method="POST" class="padding">
            {% csrf_token %} {{ noodle_form.as_p }}
            <div class="divider"></div>
            {{ formset.management_form }}
                {{ formset.as_p }}
            <button type="submit" class="btn-floating btn-large waves-light waves-effect"><i
class="large material-icons">done</i></button>
            <a href="#" onclick="window.history.back(); return false;" title="Cancel"
class="btn-floating waves-effect waves-light red"><i class="material-icons">history</i></a>

        </form>
        </div>
    </div>
</div>
```

Extending Django UserProfilePro

Django`User`。 。

`UserUserAUTH_USER_MODEL`

```
AUTH_USER_MODEL = 'myapp.MyUser'
```

`manage.py migrateAUTH_USER_MODEL`。 Djangosynamic dependecy。

`UserUSERNAME_FIELD`

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

class CustomUser(AbstractBaseUser):
     email = models.EmailField(unique=True)

     USERNAME_FIELD = 'email'
```

`AbstractBaseUserUser`。 `AbstractBaseUserUser`。

Django `manage.py createsuperuserREQUIRED_FIELDS`。 Django

```
class CustomUser(AbstractBaseUser):
    ...
    first_name = models.CharField(max_length=254)
    last_name = models.CharField(max_length=254)
    ...
    REQUIRED_FIELDS = ['first_name', 'last_name']
```

Django`is_active` `get_full_name()``get_short_name()`

```
class CustomUser(AbstractBaseUser):
    ...
    is_active = models.BooleanField(default=False)
    ...
    def get_full_name(self):
        full_name = "{0} {1}".format(self.first_name, self.last_name)
        return full_name.strip()

    def get_short_name(self):
        return self.first_name
```

`UserUserManager` Django`create_user()``create_superuser()`

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

class CustomUserManager(BaseUserManager):
    def create_user(self, email, first_name, last_name, password=None):
        if not email:
            raise ValueError('Users must have an email address')

        user = self.model(
            email=self.normalize_email(email),
        )
```

```
        user.set_password(password)
        user.first_name = first_name
        user.last_name = last_name
        user.save(using=self._db)
        return user

    def create_superuser(self, email, first_name, last_name, password):
        user = self.create_user(
            email=email,
            first_name=first_name,
            last_name=last_name,
            password=password,
        )

        user.is_admin = True
        user.is_active = True
        user.save(using=self.db)
        return user
```

User *AUTH_USER_MODEL* ◦

UserPostUser

```
from django.conf import settings
from django.db import models

class Post(models.Model):
    author = models.ForeignKey(settings.AUTH_USER_MODEL, on_delete=models.CASCADE)
```

https://riptutorial.com/zh-TW/django/topic/1209/

27:

Examples

Django。

```
def create_category(name, products):
    category = Category.objects.create(name=name)
    product_api.add_products_to_category(category, products)
    activate_category(category)
```

```
>>> create_category('clothing', ['shirt', 'trousers', 'tie'])
--------------------------------------------------------------------------
ValueError: Product 'trousers' already exists
```

。 。

create_category()。

---

django.db.transaction

[a]。

```
from django.db import transaction

@transaction.atomic
def create_category(name, products):
    category = Category.objects.create(name=name)
    product_api.add_products_to_category(category, products)
    activate_category(category)
```

```
def create_category(name, products):
    with transaction.atomic():
        category = Category.objects.create(name=name)
        product_api.add_products_to_category(category, products)
        activate_category(category)
```

。

https://riptutorial.com/zh-TW/django/topic/5555/

28:

Examples

MySQL / MariaDB

DjangoMySQL 5.5。

```
$ sudo apt-get install mysql-server libmysqlclient-dev
$ sudo apt-get install python-dev python-pip              # for python 2
$ sudo apt-get install python3-dev python3-pip            # for python 3
```

Python MySQL `mysqlclient`Django

```
$ pip install mysqlclient    # python 2 and 3
$ pip install MySQL-python   # python 2
$ pip install pymysql        # python 2 and 3
```

Django。 `default-character-set`my.cnf`/etc/mysql/mariadb.conf/*.cnf`

```
   [mysql]
   #default-character-set = latin1    #default on some systems.
   #default-character-set = utf8mb4   #default on some systems.
   default-character-set = utf8

   ...
   [mysqld]
   #character-set-server  = utf8mb4
   #collation-server      = utf8mb4_general_ci
   character-set-server  = utf8
   collation-server      = utf8_general_ci
```

MySQLMariaDB

```
#myapp/settings/settings.py

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'DB_NAME',
        'USER': 'DB_USER',
        'PASSWORD': 'DB_PASSWORD',
        'HOST': 'localhost',   # Or an IP Address that your database is hosted on
        'PORT': '3306',
        #optional:
        'OPTIONS': {
            'charset' : 'utf8',
            'use_unicode' : True,
             'init_command': 'SET '
                'storage_engine=INNODB,'
                'character_set_connection=utf8,'
                'collation_connection=utf8_bin'
```

```
                #'sql_mode=STRICT_TRANS_TABLES,'    # see note below
                #'SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED',
        },
        'TEST_CHARSET': 'utf8',
        'TEST_COLLATION': 'utf8_general_ci',
    }
}
```

OracleMySQL`ENGINE`

```
'ENGINE': 'mysql.connector.django',
```

```
CREATE DATABASE mydatabase CHARACTER SET utf8 COLLATE utf8_bin
```

> MySQL 5.7MySQL 5.6sql_mode**STRICT_TRANS_TABLES** 。 。 DjangoMySQL STRICT_TRANS_TABLESSTRICT_ALL_TABLES。 */etc/my.cnf* `sql-mode = STRICT_TRANS_TABLES`

PostgreSQL

```
sudo apt-get install libpq-dev
pip install psycopg2
```

PostgreSQL

```
#myapp/settings/settings.py

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'myprojectDB',
        'USER': 'myprojectuser',
        'PASSWORD': 'password',
        'HOST': '127.0.0.1',
        'PORT': '5432',
    }
}
```

`django.db.backends.postgresql_psycopg2` 。

Postresql

Modelfields

```
ArrayField         # A field for storing lists of data.
HStoreField        # A field for storing mappings of strings to strings.
JSONField          # A field for storing JSON encoded data.
IntegerRangeField  # Stores a range of integers
BigIntegerRangeField # Stores a big range of integers
FloatRangeField    # Stores a range of floating point values.
DateTimeRangeField # Stores a range of timestamps
```

sqliteDjango。 。

```
#myapp/settings/settings.py

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': 'db/development.sqlite3',
        'USER': '',
        'PASSWORD': '',
        'HOST': '',
        'PORT': '',
    },
}
```

◦ dumpdata

```
 ./manage.py dumpdata > databasedump.json                # full database
 ./manage.py dumpdata myapp > databasedump.json          # only 1 app
 ./manage.py dumpdata myapp.mymodel > databasedump.json  # only 1 model (table)
```

json

```
./manage.py loaddata databasedump.json
```

loadddata Djangofixtures FIXTURE_DIRS。

```
/myapp
   /fixtures
        myfixtures.json
        morefixtures.xml
```

JSON, XML or YAML

Fixtures JSON

```
[
  {
    "model": "myapp.person",
    "pk": 1,
    "fields": {
      "first_name": "John",
      "last_name": "Lennon"
    }
  },
  {
    "model": "myapp.person",
    "pk": 2,
    "fields": {
      "first_name": "Paul",
      "last_name": "McCartney"
    }
  }
]
```

YAML

```
- model: myapp.person
  pk: 1
  fields:
    first_name: John
    last_name: Lennon
- model: myapp.person
  pk: 2
  fields:
    first_name: Paul
    last_name: McCartney
```

Fixtures XML

```
<?xml version="1.0" encoding="utf-8"?>
<django-objects version="1.0">
    <object pk="1" model="myapp.person">
        <field type="CharField" name="first_name">John</field>
        <field type="CharField" name="last_name">Lennon</field>
    </object>
    <object pk="2" model="myapp.person">
        <field type="CharField" name="first_name">Paul</field>
        <field type="CharField" name="last_name">McCartney</field>
    </object>
</django-objects>
```

Django Cassandra

- pip `$ pip install django-cassandra-engine`
- settings.pyINSTALLED_APPS `INSTALLED_APPS = ['django_cassandra_engine']`
- Cange DATABASES

```
DATABASES = {
    'default': {
        'ENGINE': 'django_cassandra_engine',
        'NAME': 'db',
        'TEST_NAME': 'test_db',
        'HOST': 'db1.example.com,db2.example.com',
        'OPTIONS': {
            'replication': {
                'strategy_class': 'SimpleStrategy',
                'replication_factor': 1
            }
        }
    }
}
```

Cassandracqlsh

```
DATABASES = {
'default': {
    'ENGINE': 'django_cassandra_engine',
    'NAME': 'db',
    'TEST_NAME': 'test_db',
```

```
    'USER_NAME'='cassandradb',
    'PASSWORD'= '123cassandra',
    'HOST': 'db1.example.com,db2.example.com',
    'OPTIONS': {
        'replication': {
            'strategy_class': 'SimpleStrategy',
            'replication_factor': 1
        }
    }
}
```

}

https://riptutorial.com/zh-TW/django/topic/4933/

29:

Examples

Django`settings.py`

```
DATABASES = {
    'default': {
        'NAME': 'app_data',
        'ENGINE': 'django.db.backends.postgresql',
        'USER': 'django_db_user',
        'PASSWORD': os.environ['LOCAL_DB_PASSWORD']
    },
    'users': {
        'NAME': 'remote_data',
        'ENGINE': 'django.db.backends.mysql',
        'HOST': 'remote.host.db',
        'USER': 'remote_user',
        'PASSWORD': os.environ['REMOTE_DB_PASSWORD']
    }
}
```

`dbrouters.pyremote_data`

```
class DbRouter(object):
    """
    A router to control all database operations on models in the
    auth application.
    """
    def db_for_read(self, model, **hints):
        """
        Attempts to read remote models go to remote database.
        """
        if model._meta.app_label == 'remote':
            return 'remote_data'
        return 'app_data'

    def db_for_write(self, model, **hints):
        """
        Attempts to write remote models go to the remote database.
        """
        if model._meta.app_label == 'remote':
            return 'remote_data'
        return 'app_data'

    def allow_relation(self, obj1, obj2, **hints):
        """
        Do not allow relations involving the remote database
        """
        if obj1._meta.app_label == 'remote' or \
           obj2._meta.app_label == 'remote':
           return False
        return None

    def allow_migrate(self, db, app_label, model_name=None, **hints):
        """
```

```
        Do not allow migrations on the remote database
        """
        if model._meta.app_label == 'remote':
            return False
        return True
```

dbrouter.pysettings.py

```
DATABASE_ROUTERS = ['path.to.DbRouter', ]
```

obj.save()db_for_write◦

```
obj.save(using='other_db')
obj.delete(using='other_db')
```

```
MyModel.objects.using('other_db').all()
```

https://riptutorial.com/zh-TW/django/topic/3395/

30:

◦ Django◦

UTC◦ UTC◦

Examples

settings.pyUSE_TZ = True ◦ TIME_ZONETIME_ZONE='UTC' ◦ ◦

USE_TZFalseTIME_ZONEDjango◦ USE_TZ TIME_ZONEDjango◦

djangodatetimeUTC

Pythondatetime.datetimetzinfo◦ AwareNaive◦

.is_naive().is_aware()

settings.pyUSE_TZ settings.pysettings.pyTIME_ZONE datetime◦

◦ ◦

```
import pytz

from django.utils import timezone

# make sure you add `TimezoneMiddleware` appropriately in settings.py
class TimezoneMiddleware(object):
    """
    Middleware to properly handle the users timezone
    """

    def __init__(self, get_response):
        self.get_response = get_response

    def __call__(self, request):
        # make sure they are authenticated so we know we have their tz info.
        if request.user.is_authenticated():
            # we are getting the users timezone that in this case is stored in
            # a user's profile
            tz_str = request.user.profile.timezone
            timezone.activate(pytz.timezone(tz_str))
        # otherwise deactivate and the default time zone will be used anyway
        else:
            timezone.deactivate()

        response = self.get_response(request)
        return response
```

◦ ◦ __call__◦ ◦ timezone.activate()timezone.activate() ◦ datetimepytz.timezone(str) ◦

datetime“UTC”◦ datetime◦

```
{{ my_datetime_value }}
```

```
{% load tz %}
{% localtime on %}
    {# this time will be respect the users time zone #}
    {{ your_date_time }}
{% endlocaltime %}

{% localtime off %}
    {# this will not respect the users time zone #}
    {{ your_date_time }}
{% endlocaltime %}
```

Django 1.10。 1.10django[*MiddlewareMixin*]

https://riptutorial.com/zh-TW/django/topic/10566/

31:

PythonWebWeb。 -Django -

Examples

[]Hello World Equivalent

html“Hello World”。

1. `my_project/my_app/views.py`

```
from django.http import HttpResponse

def hello_world(request):
    html = "<html><title>Hello World!</title><body>Hello World!</body></html>"
    return HttpResponse(html)
```

2. `my_project/my_app/urls.py`url

```
from django.conf.urls import url

from . import views

urlpatterns = [
    url(r'^hello_world/$', views.hello_world, name='hello_world'),
]
```

3. `python manage.py runserver`

   `http://localhost:8000/hello_world/` html。

https://riptutorial.com/zh-TW/django/topic/7490/

32:

`QuerysetModel`。

Examples

```
class MyModel(models.Model):
    name = models.CharField(max_length=10)
    model_num = models.IntegerField()
    flag = models.NullBooleanField(default=False)
```

id / pk4
id4。

```
MyModel.objects.get(pk=4)
```

```
MyModel.objects.all()
```

`flagTrue`

```
MyModel.objects.filter(flag=True)
```

`model_num`25

```
MyModel.objects.filter(model_num__gt=25)
```

`name`“Cheap Item” `flagFalse`

```
MyModel.objects.filter(name="Cheap Item", flag=False)
```

`name`

```
MyModel.objects.filter(name__contains="ch")
```

`name`

```
MyModel.objects.filter(name__icontains="ch")
```

Q

```
class MyModel(models.Model):
    name = models.CharField(max_length=10)
    model_num = models.IntegerField()
    flag = models.NullBooleanField(default=False)
```

`QAND OR。 flag=True model_num>15 。`

```
from django.db.models import Q
MyModel.objects.filter(Q(flag=True) | Q(model_num__gt=15))
```

`WHERE flag=True OR model_num > 15`**AND**。

```
MyModel.objects.filter(Q(flag=True) & Q(model_num__gt=15))
```

`Q`~**NOT**。 `flag=False model_num!=15`

```
MyModel.objects.filter(Q(flag=True) & ~Q(model_num=15))
```

`filter()`Q“normal”`Q`。 `flagTrue15` “H”。

```
from django.db.models import Q
MyModel.objects.filter(Q(flag=True) | Q(model_num__gt=15), name__startswith="H")
```

`Qfilter exclude get` 。 `getMultipleObjectsReturned`。

ManyToManyFieldn + 1

```
# models.py:
class Library(models.Model):
    name = models.CharField(max_length=100)
    books = models.ManyToManyField(Book)

class Book(models.Model):
    title = models.CharField(max_length=100)
```

```
# views.py
def myview(request):
    # Query the database.
    libraries = Library.objects.all()

    # Query the database on each iteration (len(author) times)
    # if there is 100 librairies, there will have 100 queries plus the initial query
    for library in libraries:
        books = library.books.all()
        books[0].title
        # ...

    # total : 101 queries
```

`ManyToManyFieldManyToManyFieldprefetch_related` 。

```
# views.py
def myview(request):
    # Query the database.
    libraries = Library.objects.prefetch_related('books').all()
```

```
    # Does not query the database again, since `books` is pre-populated
    for library in libraries:
        books = library.books.all()
        books[0].title
        # ...

    # total : 2 queries - 1 for libraries, 1 for books
```

prefetch_related

```
# models.py:
class User(models.Model):
    name = models.CharField(max_length=100)

class Library(models.Model):
    name = models.CharField(max_length=100)
    books = models.ManyToManyField(Book)

class Book(models.Model):
    title = models.CharField(max_length=100)
    readers = models.ManyToManyField(User)
```

```
 # views.py
def myview(request):
    # Query the database.
    libraries = Library.objects.prefetch_related('books', 'books__readers').all()

    # Does not query the database again, since `books` and `readers` is pre-populated
    for library in libraries:
        for book in library.books.all():
            for user in book.readers.all():
                user.name
                # ...

    # total : 3 queries - 1 for libraries, 1 for books, 1 for readers
```

。

```
 # views.py
def myview(request):
    # Query the database.
    libraries = Library.objects.prefetch_related('books').all()
    for library in libraries:
        for book in library.books.filter(title__contains="Django"):
            print(book.name)
```

Django 1.7`Prefetch`

```
from django.db.models import Prefetch
# views.py
def myview(request):
    # Query the database.
    libraries = Library.objects.prefetch_related(
        Prefetch('books', queryset=Book.objects.filter(title__contains="Django")
    ).all()
    for library in libraries:
        for book in library.books.all():
```

```
            print(book.name)  # Will print only books containing Django for each library
```

ForeignKeyn + 1

Django。

```
# models.py:
class Author(models.Model):
    name = models.CharField(max_length=100)

class Book(models.Model):
    author = models.ForeignKey(Author, related_name='books')
    title = models.CharField(max_length=100)
```

```
# views.py
def myview(request):
    # Query the database
    books = Book.objects.all()

    for book in books:
        # Query the database on each iteration to get author (len(books) times)
        # if there is 100 books, there will have 100 queries plus the initial query
        book.author
        # ...

    # total : 101 queries
```

django。。

ForeignKeyForeignKeyselect_related。

```
# views.py
def myview(request):
    # Query the database.
    books = Books.objects.select_related('author').all()

    for book in books:
        # Does not query the database again, since `author` is pre-populated
        book.author
        # ...

    # total : 1 query
```

select_related

```
# models.py:
class AuthorProfile(models.Model):
    city = models.CharField(max_length=100)

class Author(models.Model):
    name = models.CharField(max_length=100)
    profile = models.OneToOneField(AuthorProfile)
```

```
class Book(models.Model):
    author = models.ForeignKey(Author, related_name='books')
    title = models.CharField(max_length=100)
```

```
# views.py
def myview(request):
    books = Book.objects.select_related('author')\
                        .select_related('author__profile').all()

    for book in books:
        # Does not query database
        book.author.name
        # or
        book.author.profile.city
        # ...

    # total : 1 query
```

Django querysetSQL

querysetquerySQL。

```
>>> queryset = MyModel.objects.all()
>>> print(queryset.query)
SELECT "myapp_mymodel"."id", ... FROM "myapp_mymodel"
```

。。SQL。

QuerySet

```
MyModel.objects.first()
```

```
MyModel.objects.last()
```

Filter First

```
MyModel.objects.filter(name='simple').first()
```

Filter Last

```
MyModel.objects.filter(name='simple').last()
```

F

F。Python。 - F

F()。 F()。。

.....

```
SomeModel(models.Model):
    ...
    some_field = models.IntegerField()
```

...idsome_fieldidF()

```
SomeModel.objects.filter(some_field=F('id') * 2)
```

F('id')id◦ DjangoSQL◦

```
SELECT * FROM some_app_some_model
WHERE some_field = ((id * 2))
```

F()SQLPython◦

-
- F
- TinyInstance

F()

◦ - F

TinyInstance◦

https://riptutorial.com/zh-TW/django/topic/1235/

33:

Python。 。 `django.db.models.Model` API。

Examples

`models.py`。 `django.db.models`。

```
from django.db import models

class Book(models.Model):
    title = models.CharField(max_length=100)
    author = models.ForeignKey('Author', on_delete=models.CASCADE,
related_name='authored_books')
    publish_date = models.DateField(null=True, blank=True)

    def __str__(self): # __unicode__ in python 2.*
        return self.title
```

。

- `title`100
- `author`ForeignKey /`Author` 。 `on_delete` `Author` 。 django 1.9 `on_delete`。 django 2 。 `CASCADE` 。
- `publish_date`。 `null`和`blank`True。

`__str__`string。

。 Django。 `manage.py`

```
python manage.py makemigrations <appname>
```

`migrations`。 `<appname>`settings.pyINSTALLED_APPS。 。

--dry-run

```
python manage.py makemigrations --dry-run
```

```
python manage.py migrate <appname>
```

。

。

```
python manage.py migrate --run-syncdb
```

Django`<appname>_<classname>`。 。 `Meta`和`db_table`

```
from django.db import models

class YourModel(models.Model):
    parms = models.CharField()
    class Meta:
        db_table = "custom_table_name"
```

SQL

```
python manage.py sqlmigrate <app_label> <migration_number>
```

Django> 1.10

makemigrations --check◦

◦

```
from django.db import models

class Author(models.Model):
   name = models.CharField(max_length=50)

#Book has a foreignkey (many to one) relationship with author
class Book(models.Model):
    author = models.ForeignKey(Author, on_delete=models.CASCADE)
    publish_date = models.DateField()
```

◦

```
class Topping(models.Model):
    name = models.CharField(max_length=50)

# One pizza can have many toppings and same topping can be on many pizzas
class Pizza(models.Model):
    name = models.CharField(max_length=50)
    toppings = models.ManyToManyField(Topping)
```

◦ ManyToManyField◦ AppointmentCustomerManyToManyField PizzaToppings◦

Through

```
class Service(models.Model):
     name = models.CharField(max_length=35)

class Client(models.Model):
    name = models.CharField(max_length=35)
    age = models.IntegerField()
    services = models.ManyToManyField(Service, through='Subscription')

class Subscription(models.Model):
     client = models.ForeignKey(Client)
     service = models.ForeignKey(Service)
     subscription_type = models.CharField(max_length=1, choices=SUBSCRIPTION_TYPES)
     created_at = models.DateTimeField(default=timezone.now)
```

◦ ◦ M2M`pizza.toppings.add(topping)` `Subscription.objects.create(client=client, service=service, subscription_type='p')`

`through tablesJoinColumn` Intersection tablemapping table

```
class Employee(models.Model):
   name = models.CharField(max_length=50)
   age = models.IntegerField()
   spouse = models.OneToOneField(Spouse)

class Spouse(models.Model):
   name = models.CharField(max_length=50)
```

◦

Django DB

Django ORMSQL◦

```
class Author(models.Model):
   name = models.CharField(max_length=50)

class Book(models.Model):
   name = models.CharField(max_length=50)
   author = models.ForeignKey(Author)
```

django`migrate`◦ Django shell

```
python manage.py shell
```

python shellDjango◦

`models.py`

```
from .models import Book, Author
```

```
>>> Author.objects.all()
[]
>>> Book.objects.all()
[]
```

```
>>> hawking = Author(name="Stephen hawking")
>>> hawking.save()
>>> history_of_time = Book(name="history of time", author=hawking)
>>> history_of_time.save()
```

create function

```
>>> wings_of_fire = Book.objects.create(name="Wings of Fire", author="APJ Abdul Kalam")
```

```
>>> Book.objects.all()
[<Book: Book object>]
>>> book = Book.objects.first() #getting the first book object
>>> book.name
u'history of time'
```

selectwhere

```
>>> Book.objects.filter(name='nothing')
[]
>>> Author.objects.filter(name__startswith='Ste')
[<Author: Author object>]
```

```
>>> book = Book.objects.first() #getting the first book object
>>> book.author.name # lookup on related model
u'Stephen hawking'
```

Stephen Hawking

```
>>> hawking.book_set.all()
[<Book: Book object>]
```

_set""Bookbook_setbook_set/。

。

Django。 Django。 Meta managed = False 。

```
class Dummy(models.Model):
    something = models.IntegerField()

    class Meta:
       managed = False
```

SQL。

```
CREATE VIEW myapp_dummy AS
SELECT id, something FROM complicated_table
WHERE some_complicated_condition = True
```

```
>>> Dummy.objects.all()
[<Dummy: Dummy object>, <Dummy: Dummy object>, <Dummy: Dummy object>]
>>> Dummy.objects.filter(something=42)
[<Dummy: Dummy object>]
```

。

```
from django.db import models
from django.urls import reverse
from django.utils.encoding import python_2_unicode_compatible
```

```
@python_2_unicode_compatible
class Book(models.Model):
    slug = models.SlugField()
    title = models.CharField(max_length=128)
    publish_date = models.DateField()

    def get_absolute_url(self):
        return reverse('library:book', kwargs={'pk':self.pk})

    def __str__(self):
        return self.title

    class Meta:
        ordering = ['publish_date', 'title']
```

`get_absolute_urlself.pk pk`。 django。 。

`get_absolute_url` 。 。 `book.get_absolute_url`。 django“”。

`__str__`。 `<a href="{{ book.get_absolute_url }}">{{ book }}</a>`。 。 。

python 2`__str____unicode__`python 3。 。

S field

slugchar。 。 url。

Meta

`Meta`。 。 ListView。 。 。

`verbose_nameverbose_name_plural` 。 。 's'。

。 Django Rest Framework。

python。

```
def expire():
    return timezone.now() + timezone.timedelta(days=7)

class Coupon(models.Model):
    expiration_date = models.DateField(default=expire)

    @property
    def is_expired(self):
        return timezone.now() > self.expiration_date
```

。

python。

python2`Model.__str__()Model.__unicode__()` ◦ `str()` `str()` ◦

1. ◦

```
# your_app/models.py

from django.db import models

class Book(models.Model):
    name = models.CharField(max_length=50)
    author = models.CharField(max_length=50)
```

2.
```
>>> himu_book = Book(name='Himu Mama', author='Humayun Ahmed')
>>> himu_book.save()
```

3. `print()`

```
>>> print(himu_book)
<Book: Book object>
```

<BookBook object> ◦ `__str__`◦

```
from django.utils.encoding import python_2_unicode_compatible

@python_2_unicode_compatible
class Book(models.Model):
    name = models.CharField(max_length=50)
    author = models.CharField(max_length=50)

    def __str__(self):
        return '{} by {}'.format(self.name, self.author)
```

python 2`python_2_unicode_compatible`◦ `__str____unicode__`◦ `django.utils.encoding`◦

print

```
>>> print(himu_book)
Himu Mama by Humayun Ahmed
```

`ForeignKeyFieldManyToManyFieldModelForm`◦

mixins

◦ ◦ **mixin** ◦

```
class PostableMixin(models.Model):
    class Meta:
        abstract=True

    sender_name = models.CharField(max_length=128)
    sender_address = models.CharField(max_length=255)
    receiver_name = models.CharField(max_length=128)
```

```
    receiver_address = models.CharField(max_length=255)
    post_datetime = models.DateTimeField(auto_now_add=True)
    delivery_datetime = models.DateTimeField(null=True)
    notes = models.TextField(max_length=500)

class Envelope(PostableMixin):
    ENVELOPE_COMMERCIAL = 1
    ENVELOPE_BOOKLET = 2
    ENVELOPE_CATALOG = 3

    ENVELOPE_TYPES = (
        (ENVELOPE_COMMERCIAL, 'Commercial'),
        (ENVELOPE_BOOKLET, 'Booklet'),
        (ENVELOPE_CATALOG, 'Catalog'),
    )

    envelope_type = models.PositiveSmallIntegerField(choices=ENVELOPE_TYPES)

class Package(PostableMixin):
    weight = models.DecimalField(max_digits=6, decimal_places=2)
    width = models.DecimalField(max_digits=5, decimal_places=2)
    height = models.DecimalField(max_digits=5, decimal_places=2)
    depth = models.DecimalField(max_digits=5, decimal_places=2)
```

`Metaabstract=True` ◦ Django◦ `EnvelopePackage` ◦

◦ mixin◦ `PostableMixin`

```
class PostableMixin(models.Model):
    class Meta:
        abstract=True

    ...
    ...

    def set_delivery_datetime(self, dt=None):
        if dt is None:
            from django.utils.timezone import now
            dt = now()

        self.delivery_datetime = dt
        self.save()
```

```
>> envelope = Envelope.objects.get(pk=1)
>> envelope.set_delivery_datetime()

>> pack = Package.objects.get(pk=1)
>> pack.set_delivery_datetime()
```

UUID

◦ 1,2,3◦

◦

UUID32ID◦ ID◦ PostgreSQLuuidchar32◦

```
import uuid
from django.db import models

class ModelUsingUUID(models.Model):
    id = models.UUIDField(primary_key=True, default=uuid.uuid4, editable=False)
```

`7778c552-73fc-4bc4-8bf9-5a2f6f7b7f47`

- “”
-

```
from django.db import models

class Place(models.Model):
    name = models.CharField(max_length=50)
    address = models.CharField(max_length=80)

class Restaurant(Place):
    serves_hot_dogs = models.BooleanField(default=False)
    serves_pizza = models.BooleanField(default=False)
```

2`Place` `Restaurant` `OneToOnePlace`。

Restaurantplaces。

https://riptutorial.com/zh-TW/django/topic/888/

34:

	true`null`
	true。 Django。
	2。 。 `[('m', 'Male'),('f','Female'),('z','Prefer Not to Disclose')]` 。 `[('Video Source',((1,'YouTube'),(2,'Facebook')),('Audio Source',((3, 'Soundcloud'),(4, 'Spotify'))]`
db_column	django。
db_index	`True`
db_tablespace	。 。
	。 。 。 lambdas。
	`False` `ModelForm` 。 `True` 。
error_messages	。 。 `null` `blank` `invalid` `invalid_choice` `uniqueunique_for_date` ;。
help_text	。 HTML。
on_delete	ForeignKeyDjangoon_deleteSQL。 `ForeignKeyOneToOneField`。 。
	`True` 。 Django;。 。
	`True` 。 。
unique_for_date	`DateFieldDateTimeField` 。
unique_for_month	`unique_for_date`。
unique_for_year	`unique_for_date`。
verbose_name	django。
	。

-
- `save()`

Examples

AutoField

。

```
from django.db import models

class MyModel(models.Model):
    pk = models.AutoField()
```

`id` 。 id。

BigIntegerField

`-9223372036854775808922337203685477580` 8 Bytes 。

```
from django.db import models

class MyModel(models.Model):
    number_of_seconds = models.BigIntegerField()
```

IntegerField

IntegerField-2147483648~2147483647 `4 Bytes` 。

```
from django.db import models

class Food(models.Model):
    name = models.CharField(max_length=255)
    calorie = models.IntegerField(default=0)
```

`default` 。 。

PositiveIntegerField

IntegerField0。 PositiveIntegerField02147483647 `4 Bytes` 。 。 。 。

```
from django.db import models

class Food(models.Model):
    name = models.CharField(max_length=255)
    calorie = models.PositiveIntegerField(default=0)
```

`default` 。 。

SmallIntegerField

SmallIntegerField-32768~32767 `2 Bytes` 。 。

```
from django.db import models
```

```
class Place(models.Model):
    name = models.CharField(max_length=255)
    temperature = models.SmallIntegerField(null=True)
```

PositiveSmallIntegerField

SmallIntegerField032767 `2 Bytes` 。 SmallIntegerField。 。

```
from django.db import models

class Staff(models.Model):
    first_name = models.CharField(max_length=255)
    last_name = models.CharField(max_length=255)
    age = models.PositiveSmallIntegerField(null=True)
```

PositiveSmallIntegerFieldEnumDjangoic

```
from django.db import models
from django.utils.translation import gettext as _

APPLICATION_NEW = 1
APPLICATION_RECEIVED = 2
APPLICATION_APPROVED = 3
APPLICATION_REJECTED = 4

APLICATION_CHOICES = (
    (APPLICATION_NEW, _('New')),
    (APPLICATION_RECEIVED, _('Received')),
    (APPLICATION_APPROVED, _('Approved')),
    (APPLICATION_REJECTED, _('Rejected')),
)

class JobApplication(models.Model):
    first_name = models.CharField(max_length=255)
    last_name = models.CharField(max_length=255)
    status = models.PositiveSmallIntegerField(
        choices=APLICATION_CHOICES,
        default=APPLICATION_NEW
    )
    ...
```

。 。

DecimalField

DecimalPython。 IntegerField2

1. *DecimalField.max_digits* 。 decimal_places。
2. *DecimalField.decimal_places* 。

993`max_digits=5`max_digits=5 `decimal_places=3`

```
class Place(models.Model):
```

```
    name = models.CharField(max_length=255)
    atmospheric_pressure = models.DecimalField(max_digits=5, decimal_places=3)
```

BinaryField

◦ ◦ base64◦

◦

```
from django.db import models

class MyModel(models.Model):
    my_binary_data = models.BinaryField()
```

CharField

CharField◦ 128◦ ◦

```
from django.db import models

class MyModel(models.Model):
    name = models.CharField(max_length=128, blank=True)
```

DateTimeField

DateTimeField◦

```
class MyModel(models.Model):
    start_time = models.DateFimeField(null=True, blank=True)
    created_on = models.DateTimeField(auto_now_add=True)
    updated_on = models.DateTimeField(auto_now=True)
```

DateTimeField

- auto_now_add◦
- auto_now◦

default◦

ForeignKey

ForeignKeymany-to-one◦ ◦

```
from django.db import models

class Person(models.Model):
    GENDER_FEMALE = 'F'
    GENDER_MALE = 'M'
```

```
    GENDER_CHOICES = (
        (GENDER_FEMALE, 'Female'),
        (GENDER_MALE, 'Male'),
    )

    first_name = models.CharField(max_length=100)
    last_name = models.CharField(max_length=100)
    gender = models.CharField(max_length=1, choices=GENDER_CHOICES)
    age = models.SmallIntegerField()


class Car(model.Model)
    owner = models.ForeignKey('Person')
    plate = models.CharField(max_length=15)
    brand = models.CharField(max_length=50)
    model = models.CharField(max_length=50)
    color = models.CharField(max_length=50)
```

◦ on_delete◦ Django 2.0◦

```
class Car(model.Model)
    owner = models.ForeignKey('Person', on_delete=models.CASCADE)
    ...
```

PersonCar◦ ◦

```
class Car(model.Model)
    owner = models.ForeignKey('Person', on_delete=models.PROTECT)
    ...
```

PersonCar◦ PersonCar◦ Person◦

https://riptutorial.com/zh-TW/django/topic/3686/

35:

/。

Examples

Queryset

```
class Product(models.Model):
    name = models.CharField(max_length=20)
    price = models.FloatField()
```

```
>>> from django.db.models import Avg, Max, Min, Sum
>>> Product.objects.all().aggregate(Avg('price'))
# {'price__avg': 124.0}
```

```
>>> Product.objects.all().aggregate(Min('price'))
# {'price__min': 9}
```

```
>>> Product.objects.all().aggregate(Max('price'))
# {'price__max':599 }
```

```
>>> Product.objects.all().aggregate(Sum('price'))
# {'price__sum':92456 }
```

```
class Category(models.Model):
    name = models.CharField(max_length=20)


class Product(models.Model):
    name = models.CharField(max_length=64)
    category = models.ForeignKey(Category, on_delete=models.PROTECT)
```

```
>>> categories = Category.objects.annotate(Count('product'))
```

```
<field_name>__count
```

```
>>> categories.values_list('name', 'product__count')
[('Clothing', 42), ('Footwear', 12), ...]
```

```
>>> categories = Category.objects.annotate(num_products=Count('product'))
```

querysets

```
>>> categories.order_by('num_products')
[<Category: Footwear>, <Category: Clothing>]
```

```
>>> categories.filter(num_products__gt=20)
[<Category: Clothing>]
```

GROUB BY ... COUNT / SUM Django ORM

Django ORM`GROUP BY ... COUNT``GROUP BY ... SUM` SQL`annotate()` `values()` `order_by()` `django.db.models``Count``Sum`

```
   class Books(models.Model):
       title  = models.CharField()
       author = models.CharField()
       price = models.FloatField()
```

`GROUP BY ... COUNT`

- `Books`

  ```
  result = Books.objects.values('author')
                        .order_by('author')
                        .annotate(count=Count('author'))
  ```

- `result` `author``count`

  ```
    author     | count
  -------------|-------
   OneAuthor   |   5
  OtherAuthor  |   2
     ...       |  ...
  ```

`GROUB BY ... SUM`

- `Books`

  ```
   result = Books.objects.values('author')
                         .order_by('author')
                         .annotate(total_price=Sum('price'))
  ```

- `result` `author``total_price`

  ```
    author     | total_price
  -------------|-------------
   OneAuthor   |    100.35
  OtherAuthor  |     50.00
      ...      |      ...
  ```

https://riptutorial.com/zh-TW/django/topic/3775/

36:

Examples

views.py

```
class UserView(TemplateView):
  """ Supply the request user object to the template """

  template_name = "user.html"

  def get_context_data(self, **kwargs):
    context = super(UserView, self).get_context_data(**kwargs)
    context.update(user=self.request.user)
    return context
```

user.html

```
<h1>{{ user.username }}</h1>

<div class="email">{{ user.email }}</div>
```

- user.username{{ user.username }}
- request.GET["search"]{{ request.GET.search }}
- users.count(){{ user.count }}

◦

```
{% if user.is_authenticated %}
  {% for item in menu %}
    <li><a href="{{ item.url }}">{{ item.name }}</a></li>
  {% endfor %}
{% else %}
  <li><a href="{% url 'login' %}">Login</a>
{% endif %}
```

{% url 'name' %}{% url 'name' %} urls.py◦

{% url 'login' %} - /accounts/login/

{% url 'user_profile' user.id %} - URL

{% url next %} -

◦

views.py

```
from django.views.generic import TemplateView
from MyProject.myapp.models import Item

class ItemView(TemplateView):
    template_name = "item.html"
```

```
    def items(self):
        """ Get all Items """
        return Item.objects.all()

    def certain_items(self):
        """ Get certain Items """
        return Item.objects.filter(model_field="certain")

    def categories(self):
        """ Get categories related to this Item """
        return Item.objects.get(slug=self.kwargs['slug']).categories.all()
```

item.html

```
{% for item in view.items %}
<ul>
    <li>{{ item }}</li>
</ul>
{% endfor %}
```

。

Itemname

```
{% for item in view.certain_items %}
<ul>
    <li>{{ item.name }}</li>
</ul>
{% endfor %}
```

```
from django.shortcuts import render

def view(request):
    return render(request, "template.html")
```

```
from django.shortcuts import render

def view(request):
    context = {"var1": True, "var2": "foo"}
    return render(request, "template.html", context=context)
```

template.html

```
<html>
{% if var1 %}
    <h1>{{ var2 }}</h1>
{% endif %}
</html>
```

Django 。 。

```
{{ "MAINROAD 3222"|lower }}    # mainroad 3222
{{ 10|add:15}}                 # 25
```

```
{{ "super"|add:"glue" }}       # superglue
{{ "A7"|add:"00" }}            # A700
{{ myDate | date:"D d M Y"}}   # Wed 20 Jul 2016
```

https://docs.djangoproject.com/en/dev/ref/templates/builtins/#ref-templates-builtins-filters。

app`templatetags`。 `__init__.py`

```
#/myapp/templatetags/filters.py
from django import template

register = template.Library()

@register.filter(name='tostring')
def to_string(value):
    return str(value)
```

```
#templates/mytemplate.html
{% load filters %}
{% if customer_id|tostring = customer %} Welcome back {% endif%}
```

```
{% for x in ""|ljust:"20" %}Hello World!{% endfor %}    # Hello World!Hello World!Hel...
{{ user.name.split|join:"_" }} ## replaces whitespace with '_'
```

。

。 “getters”。 。 。

```
class Foobar(models.Model):
    points_credit = models.IntegerField()

    def credit_points(self, nb_points=1):
        """Credit points and return the new points credit value."""
        self.points_credit = F('points_credit') + nb_points
        self.save(update_fields=['points_credit'])
        return self.points_credit
```

```
 You have {{ foobar.credit_points }} points!
```

。 。

`alters_data`True。 。

```
def credit_points(self, nb_points=1):
    """Credit points and return the new points credit value."""
    self.points_credit = F('points_credit') + nb_points
    self.save(update_fields=['points_credit'])
    return self.points_credit
credit_points.alters_data = True
```

{extends}{include}{blocks}

- **{extends}**。 `{% extends 'parent_template.html' %}`。
- **{block} {endblock}** html。。 `{% block content %} <html_code> {% endblock %}`。
- **{include}**。。 `{% include 'template_name.html' %} {% include 'template_name.html' with variable='value' variable2=8 %}`

。 Django。
3

```
project_directory
    ..
    templates
      front-page.html
      blogs.html
      blog-detail.html
```

1`base.html`

```
<html>
  <head>
  </head>

  <body>
        {% block content %}
        {% endblock %}
   </body>
</html>
```

2`blog.html`

```
{% extends 'base.html' %}

{% block content %}
    # write your blog related code here
{% endblock %}

# None of the code written here will be added to the template
```

HTML`blog.html`。 `{ % block %}`“”。。

33HTML div。 3`posts.html`。

blog.html

```
{% extends 'base.html' %}

{% block content %}
    # write your blog related code here
    {% include 'posts.html' %} # includes posts.html in blog.html file without passing any
data
    <!-- or -->
```

```
    {% include 'posts.html' with posts=postdata %} # includes posts.html in blog.html file
with passing posts data which is context of view function returns.
{% endblock %}
```

https://riptutorial.com/zh-TW/django/topic/588/

37:

Examples

◦ **01**◦

```
{{ variable|filter_name }}
{{ variable|filter_name:argument }}
```

```
{{ variable|filter_name:argument|another_filter }}
```

python

```
print(another_filter(filter_name(variable, argument)))
```

ModelQuerySet`verbose_name` ◦ `True` ◦

```
@register.filter
def verbose_name(model, plural=False):
    """Return the verbose name of a model.
    `model` can be either:
      - a Model class
      - a Model instance
      - a QuerySet
      - any object refering to a model through a `model` attribute.

    Usage:
      - Get the verbose name of an object
          {{ object|verbose_name }}
      - Get the plural verbose name of an object from a QuerySet
          {{ objects_list|verbose_name:True }}
    """
    if not hasattr(model, '_meta'):
        # handle the case of a QuerySet (among others)
        model = model.model
    opts = model._meta
    if plural:
        return opts.verbose_name_plural
    else:
        return opts.verbose_name
```

`simple_tag` ◦ ◦ “” ◦ ◦

```
{% useless 3 foo 'hello world' foo=True bar=baz.hello|capfirst %}
```

`foobaz`

```
{'foo': "HELLO", 'baz': {'hello': "world"}}
```

```
HELLO;hello world;bar:World;foo:True<br/>
```

```
HELLO;hello world;bar:World;foo:True<br/>
HELLO;hello world;bar:World;foo:True<br/>
```

33。

```
from django.utils.html import format_html_join

@register.simple_tag
def useless(repeat, *args, **kwargs):
    output = ';'.join(args + ['{}:{}'.format(*item) for item in kwargs.items()])
    outputs = [output] * repeat
    return format_html_join('\n', '{}<br/>', ((e,) for e in outputs))
```

format_html_join
HTMLoutputs。

Node

filtersimple_tag。 。

verbose_name

{% verbose_name obj %}	
{% verbose_name obj 'status' %}	“”
{% verbose_name obj plural %}	
{% verbose_name obj plural capfirst %}	
{% verbose_name obj 'foo' capfirst %}	
{% verbose_name obj field_name %}	
{% verbose_name obj 'foo'|add:'_bar' %}	“foo_bar”

pluralcapfirst“”。 'plural''capfirst' 。 {% verbose_name obj 'plural' %}“obj ”“obj.plural ”

```
@register.tag(name='verbose_name')
def do_verbose_name(parser, token):
    """
    - parser: the Parser object. We will use it to parse tokens into
              nodes such as variables, strings, ...
    - token: the Token object. We will use it to iterate each token
             of the template tag.
    """
    # Split tokens within spaces (except spaces inside quotes)
    tokens = token.split_contents()
    tag_name = tokens[0]
    try:
        # each token is a string so we need to parse it to get the actual
        # variable instead of the variable name as a string.
        model = parser.compile_filter(tokens[1])
    except IndexError:
```

```
        raise TemplateSyntaxError(
            "'{}' tag requires at least 1 argument.".format(tag_name))

    field_name = None
    flags = {
        'plural': False,
        'capfirst': False,
    }

    bits = tokens[2:]
    for bit in bits:
        if bit in flags.keys():
            # here we don't need `parser.compile_filter` because we expect
            # 'plural' and 'capfirst' flags to be actual strings.
            if flags[bit]:
                raise TemplateSyntaxError(
                    "'{}' tag only accept one occurrence of '{}' flag".format(
                        tag_name, bit)
                )
            flags[bit] = True
            continue
        if field_name:
            raise TemplateSyntaxError((
                "'{}' tag only accept one field name at most. {} is the second "
                "field name encountered."
            ).format(tag_name, bit)
        field_name = parser.compile_filter(bit)

    # VerboseNameNode is our renderer which code is given right below
    return VerboseNameNode(model, field_name, **flags)
```

```
class VerboseNameNode(Node):

    def __init__(self, model, field_name=None, **flags):
        self.model = model
        self.field_name = field_name
        self.plural = flags.get('plural', False)
        self.capfirst = flags.get('capfirst', False)

    def get_field_verbose_name(self):
        if self.plural:
            raise ValueError("Plural is not supported for fields verbose name.")
        return self.model._meta.get_field(self.field_name).verbose_name

    def get_model_verbose_name(self):
       if self.plural:
           return self.model._meta.verbose_name_plural
       else:
           return self.model._meta.verbose_name

    def render(self, context):
        """This is the main function, it will be called to render the tag.
        As you can see it takes context, but we don't need it here.
        For instance, an advanced version of this template tag could look for an
        `object` or `object_list` in the context if `self.model` is not provided.
        """
        if self.field_name:
            verbose_name = self.get_field_verbose_name()
        else:
            verbose_name = self.get_model_verbose_name()
```

```
        if self.capfirst:
            verbose_name = verbose_name.capitalize()
        return verbose_name
```

https://riptutorial.com/zh-TW/django/topic/1305/

38: Celery

CeleryDjango。 Celeryworker。 redisrabbitmqDjango ORM / db。

◦ celery`settings.pycelery_config.py`◦

```
from __future__ import absolute_import
import os
from celery import Celery
from django.conf import settings

# broker url
BROKER_URL = 'redis://localhost:6379/0'

# Indicate Celery to use the default Django settings module
os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'config.settings')

app = Celery('config')
app.config_from_object('django.conf:settings')
# if you do not need to keep track of results, this can be turned off
app.conf.update(
    CELERY_RESULT_BACKEND=BROKER_URL,
)

# This line will tell Celery to autodiscover all your tasks.py that are in your app folders
app.autodiscover_tasks(lambda: settings.INSTALLED_APPS)
```

`__init__.py`◦

```
# -*- coding: utf-8 -*-
# Not required for Python 3.
from __future__ import absolute_import

from .celery_config import app as celery_app  # noqa
```

celery workermanage.py◦

```
# pros is your django project,
celery -A proj worker -l info
```

Examples

2

1. celery `pip install celery`
2.

```
from __future__ import absolute_import, unicode_literals

from celery.decorators import task
```

```
@task
def add_number(x, y):
    return x + y
```

.delay()◦

add_number.delay(5, 10) 510add_number

delay.ready()◦

.result◦

```
async_result_object = add_number.delay(5, 10)
if async_result_object.ready():
    print(async_result_object.result)
```

Celery https://riptutorial.com/zh-TW/django/topic/5481/-celery-

39:

Examples

myblog

```
from django.conf import settings
from django.utils import timezone

class Article(models.Model):
    title = models.CharField(max_length=70)
    slug = models.SlugField(max_length=70, unique=True)
    author = models.ForeignKey(settings.AUTH_USER_MODEL, models.PROTECT)
    date_published = models.DateTimeField(default=timezone.now)
    is_draft = models.BooleanField(default=True)
    content = models.TextField()
```

Django Admin“”。

```
from django.contrib import admin
from myblog.models import Article

@admin.register(Article)
class ArticleAdmin(admin.ModelAdmin):
    pass
```

__str__()python2__unicode__() “name”。 “”。 __str__()

```
class Article(models.Model):
    def __str__(self):
        return self.title
```

“”。

◦ list_display

```
@admin.register(Article)
class ArticleAdmin(admin.ModelAdmin):
    list_display = ['__str__', 'author', 'date_published', 'is_draft']
```

list_display◦ ModelAdmin

```
from django.forms.utils import flatatt
from django.urls import reverse
from django.utils.html import format_html

@admin.register(Article)
class ArticleAdmin(admin.ModelAdmin):
    list_display = ['title', 'author_link', 'date_published', 'is_draft']

    def author_link(self, obj):
```

```
        author = obj.author
        opts = author._meta
        route = '{}_{}_change'.format(opts.app_label, opts.model_name)
        author_edit_url = reverse(route, args=[author.pk])
        return format_html(
            '<a{}>{}</a>', flatatt({'href': author_edit_url}), author.first_name)

    # Set the column name in the change list
    author_link.short_description = "Author"
    # Set the field to use when ordering using this column
    author_link.admin_order_field = 'author__firstname'
```

CSSJS

`Customer`

```
class Customer(models.Model):
    first_name = models.CharField(max_length=255)
    last_name = models.CharField(max_length=255)
    is_premium = models.BooleanField(default=False)
```

Django`first_namelast_name`

```
@admin.register(Customer)
class CustomerAdmin(admin.ModelAdmin):
    list_display = ['first_name', 'last_name', 'is_premium']
    search_fields = ['first_name', 'last_name']
```

“ *keyword* ”。“ ”

Javascriptadmin `Media`

```
@admin.register(Customer)
class CustomerAdmin(admin.ModelAdmin):
    list_display = ['first_name', 'last_name', 'is_premium']
    search_fields = ['first_name', 'last_name']

    class Media:
        #this path may be any you want,
        #just put it in your static folder
        js = ('js/admin/placeholder.js', )
```

DjangoIDjs

```
$(function () {
   $('#searchbar').attr('placeholder', 'Search by name')
})
```

`Media`CSS

```
class Media:
    css = {
        'all': ('css/admin/styles.css',)
         }
```

first_name。

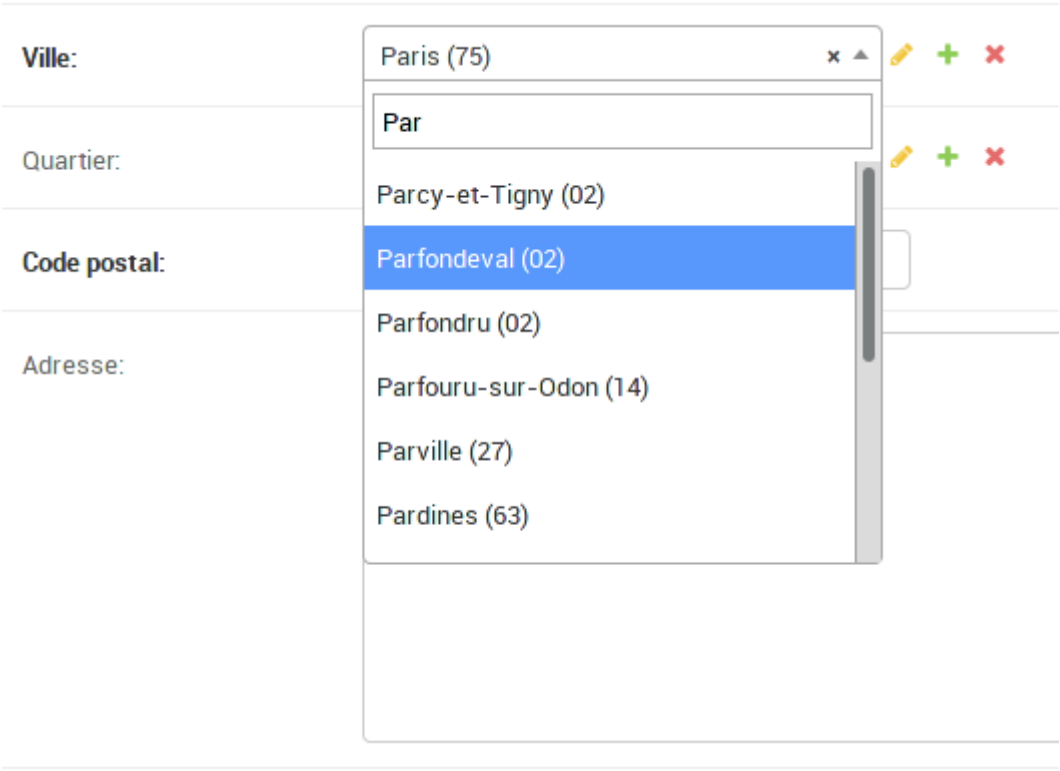
Django`list_display<td>field-'list_display_name'` css`field_first_name`

```
.field_first_name {
     background-color: #e6f2ff;
 }
```

JScssid。

Django`ForeignKey<select>。 。 。`

django-autocomplete-light DAL。 `<select>。`



views.py

```
from dal import autocomplete

class CityAutocomp(autocomplete.Select2QuerySetView):
    def get_queryset(self):
        qs = City.objects.all()
        if self.q:
            qs = qs.filter(name__istartswith=self.q)
        return qs
```

urls.py

```
urlpatterns = [
    url(r'^city-autocomp/$', CityAutocomp.as_view(), name='city-autocomp'),
```

```
]
```

forms.py

```
from dal import autocomplete

class PlaceForm(forms.ModelForm):
    city = forms.ModelChoiceField(
        queryset=City.objects.all(),
        widget=autocomplete.ModelSelect2(url='city-autocomp')
    )

    class Meta:
        model = Place
        fields = ['__all__']
```

admin.py

```
@admin.register(Place)
class PlaceAdmin(admin.ModelAdmin):
    form = PlaceForm
```

https://riptutorial.com/zh-TW/django/topic/1219/

40:

Django。

Python。

- `django-admin <command> [options]`
- `python -m django <command> [options]`
- `python manage.py <command> [options]`
- `./manage.py <command> [options]`manage.py `chmod +x manage.py`

Cron

```
*/10 * * * * pythonuser /var/www/dev/env/bin/python /var/www/dev/manage.py <command> [options]
> /dev/null
```

Examples

/Django`management commands`。

Django。

`myapp/management/commands/my_command.py`

`management``commands`__init__.py

```
from django.core.management.base import BaseCommand, CommandError

# import additional classes/modules as needed
# from myapp.models import Book

class Command(BaseCommand):
    help = 'My custom django management command'

    def add_arguments(self, parser):
       parser.add_argument('book_id', nargs='+', type=int)
       parser.add_argument('author' , nargs='+', type=str)

    def handle(self, *args, **options):
       bookid = options['book_id']
       author = options['author']
       # Your code goes here

        # For example:
        # books = Book.objects.filter(author="bob")
        # for book in books:
        #    book.name = "Bob"
        #    book.save()
```

BaseCommand。

。

```
python manage.py my_command
```

    ◦ daemonmanagement commands ◦

```
>>> python manage.py help
```

-h

```
>>> python manage.py command_name -h
```

command_name◦

```
>>> python manage.py runserver -h
>>> usage: manage.py runserver [-h] [--version] [-v {0,1,2,3}]
                              [--settings SETTINGS] [--pythonpath PYTHONPATH]
                              [--traceback] [--no-color] [--ipv6] [--nothreading]
                              [--noreload] [--nostatic] [--insecure]
                              [addrport]

Starts a lightweight Web server for development and also serves static files.

positional arguments:
  addrport              Optional port number, or ipaddr:port

optional arguments:
  -h, --help            show this help message and exit
  --version             show program's version number and exit
  -v {0,1,2,3}, --verbosity {0,1,2,3}
                        Verbosity level; 0=minimal output, 1=normal output,
                        2=verbose output, 3=very verbose output
  --settings SETTINGS   The Python path to a settings module, e.g.
                        "myproject.settings.main". If this isn't provided, the
                        DJANGO_SETTINGS_MODULE environment variable will be
                        used.
  --pythonpath PYTHONPATH
                        A directory to add to the Python path, e.g.
                        "/home/djangoprojects/myproject".
  --traceback           Raise on CommandError exceptions
  --no-color            Don't colorize the command output.
  --ipv6, -6            Tells Django to use an IPv6 address.
  --nothreading         Tells Django to NOT use threading.
  --noreload            Tells Django to NOT use the auto-reloader.
  --nostatic            Tells Django to NOT automatically serve static files
                        at STATIC_URL.
  --insecure            Allows serving static files even if DEBUG is False.
```

django-adminmanage.py

manage.pydjango-admin◦ manage.py

- PYTHONPATH
- DJANGO_SETTINGS_MODULE

```
export PYTHONPATH="/home/me/path/to/your_project"
```

```
export DJANGO_SETTINGS_MODULE="your_project.settings"
```

virtualenv`postactivate`。

`django-admin`。

Django`python manage.py [command]`manage.py+ x.`/manage.py [command]` 。

```
./manage.py help
```

localhost8000Django;

```
./manage.py runserver
```

pythonipythonDjangopython。

```
./manage.py shell
```

。

```
./manage.py makemigrations
```

。

```
./manage.py migrate
```

。

```
./manage.py test
```

`STATIC_ROOT`。

```
./manage.py collectstatic
```

。

```
./manage.py createsuperuser
```

。

```
./manage.py changepassword username
```

https://riptutorial.com/zh-TW/django/topic/1661/

41:

Examples

Querysets`as_manager`

Django mangerdjango。 djangoobjectsdjango。

I

。

- `User.objects.filter(is_active=True)` VS `User.manager.active()`
- `User.objects.filter(is_active=True).filter(is_doctor=True).filter(specialization='Dermatology')` VS `User.manager.doctors.with_specialization('Dermatology')`

psychologistsdermatologists 。

QuerySetas_managerManageras_manager 。

```
from django.db.models.query import QuerySet

class ProfileQuerySet(QuerySet):
    def doctors(self):
        return self.filter(user_type="Doctor", user__is_active=True)

    def with_specializations(self, specialization):
        return self.filter(specializations=specialization)

    def users(self):
        return self.filter(user_type="Customer", user__is_active=True)

ProfileManager = ProfileQuerySet.as_manager
```

```
class Profile(models.Model):
    ...
    manager = ProfileManager()
```

manager objects。

select_related

ForeignKey

```
from django.db import models

class Book(models.Model):
 name= models.CharField(max_length=50)
 author = models.ForeignKey(Author)
```

```
class Author(models.Model):
 name = models.CharField(max_length=50)
```

book.author.name

```
books = Book.objects.select_related('author').all()
```

。

```
class BookManager(models.Manager):

    def get_queryset(self):
        qs = super().get_queryset()
        return qs.select_related('author')

class Book(models.Model):
    ...
    objects = BookManager()
```

python 2.x`super`

```
books = Book.objects.all()
```

/。

published。

```
my_news = News.objects.filter(published=True)
```

。

```
my_news = News.objects.published()
```

。

app`managers.py` `models.Manager`

```
from django.db import models


class NewsManager(models.Manager):

    def published(self, **kwargs):
        # the method accepts **kwargs, so that it is possible to filter
        # published news
        # i.e: News.objects.published(insertion_date__gte=datetime.now)
        return self.filter(published=True, **kwargs)
```

objects

```
from django.db import models

# import the created manager
from .managers import NewsManager

class News(models.Model):
    """ News model
    """
    insertion_date = models.DateTimeField('insertion date', auto_now_add=True)
    title = models.CharField('title', max_length=255)
    # some other fields here
    published = models.BooleanField('published')

    # assign the manager class to the objects property
    objects = NewsManager()
```

```
my_news = News.objects.published()
```

```
my_news = News.objects.published(title__icontains='meow')
```

https://riptutorial.com/zh-TW/django/topic/1400/

42: Jenkins

Examples

Jenkins 2.0+

Jenkins2.x“Build Pipeline Plugin”CI/。

“”“Github“GitHub”。

Djangopython。

```
#!/usr/bin/groovy

node {
  // If you are having issues with your project not getting updated,
  // try uncommenting the following lines.
  //stage 'Checkout'
  //checkout scm
  //sh 'git submodule update --init --recursive'

  stage 'Update Python Modules'
  // Create a virtualenv in this folder, and install or upgrade packages
  // specified in requirements.txt; https://pip.readthedocs.io/en/1.1/requirements.html
  sh 'virtualenv env && source env/bin/activate && pip install --upgrade -r requirements.txt'

  stage 'Test'
  // Invoke Django's tests
  sh 'source env/bin/activate && python ./manage.py runtests'
}
```

Jenkins 2.0+Docker

Docker。 `manage.pyruntestsinvoke` / `fabric` 。

```
#!/usr/bin/groovy

node {
  stage 'Checkout'
  checkout scm
  sh 'git submodule update --init --recursive'

  imageName = 'mycontainer:build'
  remotes = [
    'dockerhub-account',
  ]

  stage 'Build'
  def djangoImage = docker.build imageName

  stage 'Run Tests'
  djangoImage.run('', 'runtests')
```

```
  stage 'Push'
  for (int i = 0; i < remotes.size(); i++) {
      sh "docker tag ${imageName} ${remotes[i]}/${imageName}"
      sh "docker push ${remotes[i]}/${imageName}"
   }
}
```

Jenkins https://riptutorial.com/zh-TW/django/topic/5873/jenkins

43:

Examples

1. - `pip install django-celery`
2.
3. ◦

```
- src/
  - bin/celery_worker_start # will be explained later on
  - logs/celery_worker.log
  - stack/__init __.py
  - stack/celery.py
  - stack/settings.py
  - stack/urls.py
  - manage.py
```

4. `celery.pystack/stack/`◦

```
from __future__ import absolute_import
import os
from celery import Celery
os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'stack.settings')
from django.conf import settings  # noqa
app = Celery('stack')
app.config_from_object('django.conf:settings')
app.autodiscover_tasks(lambda: settings.INSTALLED_APPS)
```

5. `stack/stack/__init__.py`

```
from __future__ import absolute_import
from .celery import app as celery_app  # noqa
```

6. `@shared_task()`

```
@shared_task()
def add(x, y):
    print("x*y={}".format(x*y))
```

7. “”

```
celery -A stack worker -l info
```

1. ◦ ◦ `stack/bin/celery_worker_start`

```
#!/bin/bash

NAME="StackOverflow Project - celery_worker_start"
```

```
PROJECT_DIR=/home/stackoverflow/apps/proj/proj/
ENV_DIR=/home/stackoverflow/apps/proj/env/

echo "Starting $NAME as `whoami`"

# Activate the virtual environment
cd "${PROJECT_DIR}"

if [ -d "${ENV_DIR}" ]
then
    . "${ENV_DIR}bin/activate"
fi

celery -A stack --loglevel='INFO'
```

2. `chmod u+x bin/celery_worker_start`
3. `apt-get install supervisor`
4. ◦ `/etc/supervisor/conf.d/stack_supervisor.conf`

```
[program:stack-celery-worker]
command = /home/stackoverflow/apps/stack/src/bin/celery_worker_start
user = polsha
stdout_logfile = /home/stackoverflow/apps/stack/src/logs/celery_worker.log
redirect_stderr = true
environment = LANG = en_US.UTF-8,LC_ALL = en_US.UTF-8
numprocs = 1
autostart = true
autorestart = true
startsecs = 10
stopwaitsecs = 600
priority = 998
```

5. ```
   sudo supervisorctl reread
       stack-celery-worker: available
   sudo supervisorctl update
       stack-celery-worker: added process group
   ```

6. ```
   sudo supervisorctl status stack-celery-worker
       stack-celery-worker      RUNNING    pid 18020, uptime 0:00:50
   sudo supervisorctl stop stack-celery-worker
       stack-celery-worker: stopped
   sudo supervisorctl start stack-celery-worker
       stack-celery-worker: started
   sudo supervisorctl restart stack-celery-worker
       stack-celery-worker: stopped
       stack-celery-worker: started
   ```

+ RabbitMQ

Celery。RabbitMQ。

rabbitmq

```
sudo apt-get install rabbitmq-server
```

。

```
sudo rabbitmqctl add_user myuser mypassword
sudo rabbitmqctl add_vhost myvhost
sudo rabbitmqctl set_user_tags myuser mytag
sudo rabbitmqctl set_permissions -p myvhost myuser ".*" ".*" ".*"
```

```
sudo rabbitmq-server
```

pip

```
pip install celery
```

Django settings.pyURL

```
BROKER_URL = 'amqp://myuser:mypassword@localhost:5672/myvhost'
```

```
celery -A your_app worker -l info
```

Celerydjango。

SupervisorPythonunix。。。

```
sudo apt-get install supervisor
```

supervisor conf.d/etc/supervisor/conf.d/your_proj.confyour_proj.conf

```
[program:your_proj_celery]
command=/home/your_user/your_proj/.venv/bin/celery --app=your_proj.celery:app worker -l info
directory=/home/your_user/your_proj
numprocs=1
stdout_logfile=/home/your_user/your_proj/logs/celery-worker.log
stderr_logfile=/home/your_user/your_proj/logs/low-worker.log
autostart=true
autorestart=true
startsecs=10
```

supervisorctlSupervisor。Supervisor/etc/supervisor/conf.d

```
sudo supervisorctl reread
```

```
sudo supervisorctl update
```

。

```
sudo supervisorctl status
```

```
sudo supervisorctl restart your_proj_celery
```

https://riptutorial.com/zh-TW/django/topic/7091/

44:

Examples

◦ <input type="tel"> TextInputinput_type'tel' ◦

```
from django.forms.widgets import TextInput

class PhoneInput(TextInput):
    input_type = 'tel'
```

MultiWidget◦

```
from datetime import date

from django.forms.widgets import MultiWidget, Select
from django.utils.dates import MONTHS

class SelectMonthDateWidget(MultiWidget):
    """This widget allows the user to fill in a month and a year.

    This represents the first day of this month or, if `last_day=True`, the
    last day of this month.
    """

    default_nb_years = 10

    def __init__(self, attrs=None, years=None, months=None, last_day=False):
        self.last_day = last_day

        if not years:
            this_year = date.today().year
            years = range(this_year, this_year + self.default_nb_years)
        if not months:
            months = MONTHS

        # Here we will use two `Select` widgets, one for months and one for years
        widgets = (Select(attrs=attrs, choices=months.items()),
                   Select(attrs=attrs, choices=((y, y) for y in years)))
        super().__init__(widgets, attrs)

    def format_output(self, rendered_widgets):
        """Concatenates rendered sub-widgets as HTML"""
        return (
            '<div class="row">'
            '<div class="col-xs-6">{}</div>'
            '<div class="col-xs-6">{}</div>'
            '</div>'
        ).format(*rendered_widgets)

    def decompress(self, value):
        """Split the widget value into subwidgets values.
        We expect value to be a valid date formated as `%Y-%m-%d`.
        We extract month and year parts from this string.
        """
```

```
        if value:
            value = date(*map(int, value.split('-')))
            return [value.month, value.year]
        return [None, None]

    def value_from_datadict(self, data, files, name):
        """Get the value according to provided `data` (often from `request.POST`)
        and `files` (often from `request.FILES`, not used here)
        `name` is the name of the form field.

        As this is a composite widget, we will grab multiple keys from `data`.
        Namely: `field_name_0` (the month) and `field_name_1` (the year).
        """
        datalist = [
            widget.value_from_datadict(data, files, '{}_{}'.format(name, i))
            for i, widget in enumerate(self.widgets)]
        try:
            # Try to convert it as the first day of a month.
            d = date(day=1, month=int(datelist[0]), year=int(datelist[1]))
            if self.last_day:
                # Transform it to the last day of the month if needed
                if d.month == 12:
                    d = d.replace(day=31)
                else:
                    d = d.replace(month=d.month+1) - timedelta(days=1)
        except (ValueError, TypeError):
            # If we failed to recognize a valid date
            return ''
        else:
            # Convert it back to a string with format `%Y-%m-%d`
            return str(d)
```

https://riptutorial.com/zh-TW/django/topic/1230/

45:

- NewFormSet = formset_factorySomeFormextra = 2
- formset = NewFormSetinitial = [{'some_field''Field Value''other_field''Other Field Value'}]

Examples

Formset◦ ChoiceForm◦

appname/forms.py

```
from django import forms
class ChoiceForm(forms.Form):
    choice = forms.CharField()
    pub_date = forms.DateField()
```

formset_factoryFormChoiceFormextra /formset◦

formsetextra + 1formsetinitialized + extra extra◦

appname/views.py

```
import datetime
from django.forms import formset_factory
from appname.forms import ChoiceForm
    ChoiceFormSet = formset_factory(ChoiceForm, extra=2)
    formset = ChoiceFormSet(initial=[
      {'choice': 'Between 5-15 ?',
        'pub_date': datetime.date.today(),}
      ])
```

formset objectformset object printform.as_table

Output in rendered template

```
<tr>
<th><label for="id_form-0-choice">Choice:</label></th>
<td><input type="text" name="form-0-choice" value="Between 5-15 ?" id="id_form-0-choice"
/></td>
</tr>
<tr>
<th><label for="id_form-0-pub_date">Pub date:</label></th>
<td><input type="text" name="form-0-pub_date" value="2008-05-12" id="id_form-0-pub_date"
/></td>
</tr>
<tr>
<th><label for="id_form-1-choice">Choice:</label></th>
<td><input type="text" name="form-1-choice" id="id_form-1-choice" /></td>
</tr>
<tr>
<th><label for="id_form-1-pub_date">Pub date:</label></th>
<td><input type="text" name="form-1-pub_date" id="id_form-1-pub_date" /></td
</tr>
```

```
<tr>
<th><label for="id_form-2-choice">Choice:</label></th>
<td><input type="text" name="form-2-choice" id="id_form-2-choice" /></td>
</tr>
<tr>
<th><label for="id_form-2-pub_date">Pub date:</label></th>
<td><input type="text" name="form-2-pub_date" id="id_form-2-pub_date" /></td>
</tr>
```

https://riptutorial.com/zh-TW/django/topic/6082/

46:

Examples

Syslog

Djangosyslog。 pythonSysLogHandler 。

```
from logging.handlers import SysLogHandler
LOGGING = {
    'version': 1,
    'disable_existing_loggers': True,
    'formatters': {
        'standard': {
            'format' : "[YOUR PROJECT NAME] [%(asctime)s] %(levelname)s [%(name)s:%(lineno)s]
%(message)s",
            'datefmt' : "%d/%b/%Y %H:%M:%S"
        }
    },
    'handlers': {
        'console': {
            'class': 'logging.StreamHandler',
        },
        'syslog': {
            'class': 'logging.handlers.SysLogHandler',
            'formatter': 'standard',
            'facility': 'user',
            # uncomment next line if rsyslog works with unix socket only (UDP reception
disabled)
            #'address': '/dev/log'
        }
    },
    'loggers': {
        'django':{
            'handlers': ['syslog'],
            'level': 'INFO',
            'disabled': False,
            'propagate': True
        }
    }
}

# loggers for my apps, uses INSTALLED_APPS in settings
# each app must have a configured logger
# level can be changed as desired: DEBUG, INFO, WARNING...
MY_LOGGERS = {}
for app in INSTALLED_APPS:
    MY_LOGGERS[app] = {
        'handlers': ['syslog'],
        'level': 'DEBUG',
        'propagate': True,
    }
LOGGING['loggers'].update(MY_LOGGERS)
```

Django

DjangoPython。 。

```
LOGGING = {
    'version': 1,
    'disable_existing_loggers': False,
    'formatters': {
        'default': {
            'format': "[%(asctime)s] %(levelname)s [%(name)s:%(lineno)s] %(message)s",
            'datefmt': "%Y-%m-%d %H:%M:%S"
        },
    },
    'handlers': {
        'console': {
            'level': 'INFO',
            'class': 'logging.StreamHandler',
            'formatter': 'default'
        },
    },
    'loggers': {
        'django': {
            'handlers': ['console'],
            'propagate': True,
            'level': 'INFO',
        },
    }
}
```

。 。 。

。 stdoutstderr。

```
'rotated_logs': {
    'class': 'logging.handlers.RotatingFileHandler',
    'filename': '/var/log/my_project.log',
    'maxBytes': 1024 * 1024 * 5,  # 5 MB
    'backupCount': 5,
    'formatter': 'default'
    'level': 'DEBUG',
},
```

`filenamefilename`。 5 MBmy_project.log.15。

```
'mail_admins': {
    'level': 'ERROR',
    'class': 'django.utils.log.AdminEmailHandler'
},
```

eamil`ADMINS`。 `ERROR` `ERROR`。 50x。

Django。 。 。 。

`LOGGING` 。

https://riptutorial.com/zh-TW/django/topic/1231/

47:

Examples

`settings.py` `settings.py` Django。

```
TIME_ZONE = 'UTC'  # use this, whenever possible
TIME_ZONE = 'Europe/Berlin'
TIME_ZONE = 'Etc/GMT+1'
```

Windows。

Django

```
USE_TZ = False
```

Django`UTC`

UTC。DST。DST。。

https://docs.djangoproject.com/en/stable/topics/i18n/timezones/

。

```
from django.conf import settings
```

`settings`

```
if not settings.DEBUG:
    email_user(user, message)
```

BASE_DIR

。URL。

```
import os
BASE_DIR = os.path.dirname(os.path.dirname(__file__))
```

`BASE_DIR`。

```
TEMPLATE_PATH = os.path.join(BASE_DIR, "templates")
STATICFILES_DIRS = [
    os.path.join(BASE_DIR, "static"),

]
```

。。

os.path。 project.settings.dev

```
BASE_DIR = os.path.dirname(os.path.dirname(os.path.dirname(__file__)))
```

unipathpip install unipath 。

```
from unipath import Path

BASE_DIR = Path(__file__).ancestor(2)  # or ancestor(3) if using a submodule

TEMPLATE_PATH = BASE_DIR.child('templates')
STATICFILES_DIRS = [
    BASE_DIR.child('static'),
]
```

The Twelve-Factor App 。

。

Django`settings.py`。 PythonPython`os`。

settings.py

```
import os

SECRET_KEY = os.environ.get('APP_SECRET_KEY', 'unsafe-secret-key')

DEBUG = bool(os.environ.get('DJANGO_DEBUG', True) == 'False')

ALLOWED_HOSTS = os.environ.get('DJANGO_ALLOWED_HOSTS', '').split()

DATABASES = {
    'default': {
        'ENGINE': os.environ.get('APP_DB_ENGINE', 'django.db.backends.sqlite3'),
        'NAME': os.environ.get('DB_NAME', 'db.sqlite'),
        'USER': os.environ.get('DB_USER', ''),
        'PASSWORD': os.environ.get('DB_PASSWORD', ''),
        'HOST': os.environ.get('DB_HOST', None),
        'PORT': os.environ.get('DB_PORT', None),
        'CONN_MAX_AGE': 600,
    }
}
```

Djangosqlite3。

dev / prod。 - /

DATABASE_URL 。

Django`settings.py` 。

```
myprojectroot/
```

```
    myproject/
        __init__.py
        settings/
            __init__.py
            base.py
            dev.py
            prod.py
            tests.py
```

。

settings.pysettings/base.pyfrom .base import *“” settings/base.py。 settings/dev.py

```
# -*- coding: utf-8 -*-
from .base import *  # noqa

DEBUG = True
INSTALLED_APPS.extend([
    'debug_toolbar',
])
EMAIL_BACKEND = 'django.core.mail.backends.console.EmailBackend'
INTERNAL_IPS = ['192.168.0.51', '192.168.0.69']
```

1

django-adminDJANGO_SETTINGS_MODULEmyproject.settings。 myproject.settings.dev。 myproject.settings.prod。 virtualenvpostactivate

```
#!/bin/sh
export PYTHONPATH="/home/me/django_projects/myproject:$VIRTUAL_ENV/lib/python3.4"
export DJANGO_SETTINGS_MODULE="myproject.settings.dev"
```

DJANGO_SETTINGS_MODULEdjango-admin--settings

```
django-admin test --settings=myproject.settings.tests
```

2

DJANGO_SETTINGS_MODULE myproject.settings __init__.pysettings。

__init__.py

```
from .dev import *
```

。 。

```
djangoproject
```

```
├── config
│   ├── __init__.py
│   ├── requirements
│   │   ├── base.txt
│   │   ├── dev.txt
│   │   ├── test.txt
│   │   └── prod.txt
│   └── settings
└── manage.py
```

`base.txt`◦

```
# base.txt
Django==1.8.0
psycopg2==2.6.1
jinja2==2.8
```

`-r base.txt-r base.txt` ◦

```
# dev.txt
-r base.txt # includes all dependencies in `base.txt`

# specific dependencies only used in dev env
django-queryinspect==0.1.0
```

```
# test.txt
-r base.txt # includes all dependencies in `base.txt`

# specific dependencies only used in test env
nose==1.3.7
django-nose==1.4
```

```
# prod.txt
-r base.txt # includes all dependencies in `base.txt`

# specific dependencies only used in production env
django-queryinspect==0.1.0
gunicorn==19.3.0
django-storages-redux==1.3
boto==2.38.0
```

◦ dev env `pip install -r config/requirements/dev.txt`

JSON

GitSVNVCS◦

`SECRET_KEY`◦

`secrets.json` *“ DjangoScoops ”*

```
{
    "SECRET_KEY": "N4HE:AMk:.Ader5354DR453TH8SHTQr",
```

```
    "DB_PASSWORD": "v3ry53cr3t"
}
```

`.gitignore` for git

```
*.py[co]
*.sw[po]
*~
/secrets.json
```

`settings`

```
import json
import os
from django.core.exceptions import ImproperlyConfigured

with open(os.path.join(BASE_DIR, 'secrets.json')) as secrets_file:
    secrets = json.load(secrets_file)

def get_secret(setting, secrets=secrets):
    """Get secret setting or fail with ImproperlyConfigured"""
    try:
        return secrets[setting]
    except KeyError:
        raise ImproperlyConfigured("Set the {} setting".format(setting))
```

```
SECRET_KEY = get_secret('SECRET_KEY')
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgres',
        'NAME': 'db_name',
        'USER': 'username',
        'PASSWORD': get_secret('DB_PASSWORD'),
    },
}
```

DjangoDjango 1.8Daniel Roy GreenfeldAudrey RoyGreenfeld。2015 Two Scoops PressISBN 978-0981467344

DATABASE_URL

HerokuPaaSURL.....。

`dj_database_url`DATABASE_URLDjangoPython。

```
import dj_database_url

if os.environ.get('DATABASE_URL'):
    DATABASES['default'] =
        dj_database_url.config(default=os.environ['DATABASE_URL'])
```

https://riptutorial.com/zh-TW/django/topic/942/

48:

PDB

PdbPdb`globals()` `locals()` `globals()`。

Examples

PythonPdb

Djangopdb Python。

`views.py`

```
from django.http import HttpResponse


def index(request):
    foo = 1
    bar = 0

    bug = foo/bar

    return HttpResponse("%d goes here." % bug)
```

Console

```
python manage.py runserver
```

Django`ZeroDivisionError`。

```
from django.http import HttpResponse

# Pdb import
import pdb


def index(request):
    foo = 1
    bar = 0

    # This is our new breakpoint
    pdb.set_trace()

    bug = foo/bar

    return HttpResponse("%d goes here." % bug)
```

pdb

```
python -m pdb manage.py runserver
```

shellPdb。

pdb shell

shell

```
> ../views.py(12)index()
-> bug = foo/bar
# input 'foo/bar' expression to see division results:
(Pdb) foo/bar
*** ZeroDivisionError: division by zero
# input variables names to check their values:
(Pdb) foo
1
(Pdb) bar
0
# 'bar' is a source of the problem, so if we set it's value > 0...
(Pdb) bar = 1
(Pdb) foo/bar
1.0
# exception gone, ask pdb to continue execution by typing 'c':
(Pdb) c
[03/Aug/2016 10:50:45] "GET / HTTP/1.1" 200 111
```

OK。

pdbshell`q`。

Django

django-debug-toolbar

```
pip install django-debug-toolbar
```

settings.py

- settings.py

```
# If environment is dev...
DEBUG = True

INSTALLED_APPS += [
    'debug_toolbar',
]

MIDDLEWARE += ['debug_toolbar.middleware.DebugToolbarMiddleware']
```

```
INSTALLED_APPS = [
    # ...
    'django.contrib.staticfiles',
    # ...
]

STATIC_URL = '/static/'
```

```
# If environment is dev...
DEBUG = True

INSTALLED_APPS += [
    'debug_toolbar',
]
```

settings.pyINTERNAL_IPS

```
INTERNAL_IPS = ('127.0.0.1', )
```

urls.py

urls.py

```
if settings.DEBUG and 'debug_toolbar' in settings.INSTALLED_APPS:
    import debug_toolbar
    urlpatterns += [
        url(r'^__debug__/', include(debug_toolbar.urls)),
    ]
```

```
python manage.py collectstatic
```

SQL。

HTML

django-debug-toolbartext/html <html><body>。

“”。

“”

```
assert False, value
```

django`AssertionError`。

DEBUG=True。

。

- 。 PythonDjango - 。
- 。 PEP 257GooglePython。
- **Logging** - 。
- **`assert`** 。

doctests

https://riptutorial.com/zh-TW/django/topic/5072/

49:

Examples

Django`username`password◦ `email`password◦

```
from django.contrib.auth import get_user_model

class EmailBackend(object):
    """
    Custom Email Backend to perform authentication via email
    """
    def authenticate(self, username=None, password=None):
        user_model = get_user_model()
        try:
            user = user_model.objects.get(email=username)
            if user.check_password(password): # check valid password
                return user # return user to be authenticated
        except user_model.DoesNotExist: # no matching user exists
            return None

    def get_user(self, user_id):
        user_model = get_user_model()
        try:
            return user_model.objects.get(pk=user_id)
        except user_model.DoesNotExist:
            return None
```

`AUTHENTICATION_BACKENDS`◦

```
# settings.py
AUTHENTICATION_BACKENDS = (
    'my_app.backends.EmailBackend',
    ...
    )
```

https://riptutorial.com/zh-TW/django/topic/1282/

50:

◦

◦ ◦

◦

◦ ◦ ◦

◦ ◦ ◦ ◦

◦

Examples

◦ 15

```
# imports
from django.shortcuts import render_to_response
from django.http import HttpResponseRedirect

from .models import SampleObject
from .forms import SampleObjectForm

# view functioon
def create_object(request):

    # when request method is 'GET', show the template
    if request.method == GET:
        # perform actions, such as loading a model form
        form = SampleObjectForm()
        return render_to_response('template.html', locals())

    # if request method is 'POST', create the object and redirect
    if request.method == POST:
        form = SampleObjectForm(request.POST)

        # save object and redirect to success page if form is valid
        if form.is_valid:
            form.save()
            return HttpResponseRedirect('url_to_redirect_to')

        # load template with form and show errors
        else:
            return render_to_response('template.html', locals())
```

“”◦ 7

```
from django.views.generic import CreateView

from .models import SampleObject
```

```
from .forms import SampleObjectForm

class CreateObject(CreateView):
    model = SampleObject
    form_class = SampleObjectForm
    success_url = 'url_to_redirect_to'
```

。 。

。 。 -

```
def create_object(request):
    page_title = 'My Page Title'

    # ...

    return render_to_response('template.html', locals())
```

。 。 *get_context_data*

```
class CreateObject(CreateView):
    model = SampleObject
    form_class = SampleObjectForm
    success_url = 'url_to_redirect_to'

    def get_context_data(self, **kwargs):

        # Call class's get_context_data method to retrieve context
        context = super().get_context_data(**kwargs)

        context['page_title'] = 'My page title'
        return context
```

- 。

Mixins

Mixins。 mixin。

“page_title”。get_context_datamixinmixin。

```
# Your Mixin
class CustomMixin(object):

    def get_context_data(self, **kwargs):

        # Call class's get_context_data method to retrieve context
        context = super().get_context_data(**kwargs)

        context['page_title'] = 'My page title'
        return context

# Your view function now inherits from the Mixin
class CreateObject(CustomMixin, CreateView):
    model = SampleObject
```

```
    form_class = SampleObjectForm
    success_url = 'url_to_redirect_to'

# As all other view functions which need these methods
class EditObject(CustomMixin, EditView):
    model = SampleObject
    # ...
```

◦ ◦

https://riptutorial.com/zh-TW/django/topic/9452/

51:

django-admin	
makemigrations <my_app>	my_app
makemigrations	
makemigrations --merge	
makemigrations --merge <my_app>	my_app
makemigrations --name <migration_name> <my_app>	migration_namemy_appmigration_name
migrate <my_app>	my_app
migrate	
migrate <my_app> <migration_name>	migration_name
migrate <my_app> zero	my_app
sqlmigrate <my_app> <migration_name>	SQL
showmigrations	
showmigrations <my_app>	my_app

Examples

Django。django。

```
$ django-admin makemigrations <app_name>
```

app_namemigration。 0001_initial.py 0002_ 0003 ……

<app_name>INSTALLED_APPS。

```
$ django-admin migrate <app_name>
```

```
$ django-admin showmigrations app_name
app_name
  [X] 0001_initial
  [X] 0002_auto_20160115_1027
  [X] 0003_somemodel
  [ ] 0004_auto_20160323_1826
```

- [X]
- [ ]。 django-admin migrate

migrate command 。 django-admin showmigrations

```
$ django-admin migrate app_name 0002  # Roll back to migration 0002
$ django-admin showmigrations app_name
app_name
  [X] 0001_initial
  [X] 0002_auto_20160115_1027
  [ ] 0003_somemodel
  [ ] 0004_auto_20160323_1826
```

Django。 。

```
class Article(models.Model):
    title = models.CharField(max_length=70)
```

SlugField

```
class Article(models.Model):
    title = models.CharField(max_length=70)
    slug = models.SlugField(max_length=70)
```

titleslug。

```
$ django-admin shell
>>> from my_app.models import Article
>>> from django.utils.text import slugify
>>> for article in Article.objects.all():
...     article.slug = slugify(article.title)
...     article.save()
...
>>>
```

.....。 。

。

```
$ django-admin makemigrations --empty app_name
```

。 。 0023_article_slug 0024_auto_20160719_1734 。

```
# -*- coding: utf-8 -*-
# Generated by Django 1.9.7 on 2016-07-19 15:34
from __future__ import unicode_literals

from django.db import migrations
from django.utils.text import slugify


def gen_slug(apps, schema_editor):
    # We can't import the Article model directly as it may be a newer
    # version than this migration expects. We use the historical version.
    Article = apps.get_model('app_name', 'Article')
    for row in Article.objects.all():
        row.slug = slugify(row.name)
        row.save()
```

```
class Migration(migrations.Migration):

    dependencies = [
        ('hosting', '0023_article_slug'),
    ]

    operations = [
        migrations.RunPython(gen_slug, reverse_code=migrations.RunPython.noop),
        # We set `reverse_code` to `noop` because we cannot revert the migration
        # to get it back in the previous state.
        # If `reverse_code` is not given, the migration will not be reversible,
        # which is not the behaviour we expect here.
    ]
```

Djangodjango_migrations。

。 。

```
python manage.py makemigrations your_app_label
```

```
python manage.py migrate --fake-initial
```

```
python manage.py migrate --fake
```

```
python manage.py migrate --fake core
```

```
python manage.py migrate myapp migration_name
```

makemigrations --name <your_migration_name>。

```
python manage.py makemigrations --name <your_migration_name> <app_name>
```

。 。

。 Reporternameaddress 。

Reporter。 Aage age0002_reporter_age.py。 Bbank_account0002_reporter_bank_account 。 。

Reporter 。 0002。

。

1. --mergemakemigrations。

   ```
   python manage.py makemigrations --merge <my_app>
   ```

   。

2.

◦ makemigrations◦ migrations.RunPython ◦

CharFieldForeignKey

discography

```
from django.db import models

class Album(models.Model):
    name = models.CharField(max_length=255)
    artist = models.CharField(max_length=255)
```

ForeignKey◦ ◦

1ForeignKeynull

```
from django.db import models

class Album(models.Model):
    name = models.CharField(max_length=255)
    artist = models.CharField(max_length=255)
    artist_link = models.ForeignKey('Artist', null=True)

class Artist(models.Model):
    name = models.CharField(max_length=255)
```

...◦

```
./manage.py makemigrations discography
```

2◦ ◦

```
./manage.py makemigrations --empty --name transfer_artists discography
```

RunPython◦

```
def link_artists(apps, schema_editor):
    Album = apps.get_model('discography', 'Album')
    Artist = apps.get_model('discography', 'Artist')
    for album in Album.objects.all():
        artist, created = Artist.objects.get_or_create(name=album.artist)
        album.artist_link = artist
        album.save()
```

artist_link◦ ◦

“ artist ◦ artist_linkartist ◦

Django◦

https://riptutorial.com/zh-TW/django/topic/1200/

52:

Examples

GunicornDjango

1. gunicorn

   ```
   pip install gunicorn
   ```

2. djangomanage.pygunicorndjango

   ```
   gunicorn [projectname].wsgi:application -b 127.0.0.1:[port number]

   --env

   gunicorn --env DJANGO_SETTINGS_MODULE=[projectname].settings [projectname].wsgi

   -D
   ```

3. gunicorn

   ```
   Starting gunicorn 19.5.0

   Listening at: http://127.0.0.1:[port number] ([pid])
   ```

   .... gunicorn

Heroku

1. Heroku Toolbelt 。
2. Django。 tk
3. `heroku create [app_name]` 。 Heroku。 `http://[app name].herokuapp.com`
4. `Procfile`。 。

   ```
   web: <bash command to start production server>
   ```

   - `worker-name: <bash command to start worker>`

5. requirements.txt。

- `pip freeze > requirements.txt`
- 。 Python。

6. 1. `git push heroku master`

   Herokugitdropbox。 `heroku.com`GitHub。

2. `heroku ps:scale web=1`

   web“dynos”。 dynos。

3. `heroku openhttp://app-name.herokuapp.com`

   `heroku open`herokuURL。

7. 。 DjangoHeroku“”。 Heroku。

fabfile.py

FabricPython2.5-2.7SSH。 Python。

```
pip install fabric
```

```
fabfile.py
```

```
#myproject/fabfile.py
from fabric.api import *

@task
def dev():
    # details of development server
    env.user = # your ssh user
    env.password = #your ssh password
    env.hosts = # your ssh hosts (list instance, with comma-separated hosts)
    env.key_filename = # pass to ssh key for github in your local keyfile

@task
def release():
    # details of release server
    env.user = # your ssh user
    env.password = #your ssh password
    env.hosts = # your ssh hosts (list instance, with comma-separated hosts)
    env.key_filename = # pass to ssh key for github in your local keyfile

@task
def run():
    with cd('path/to/your_project/'):
        with prefix('source ../env/bin/activate'):
        # activate venv, suppose it appear in one level higher
            # pass commands one by one
            run('git pull')
            run('pip install -r requirements.txt')
            run('python manage.py migrate --noinput')
            run('python manage.py collectstatic --noinput')
            run('touch reload.txt')
```

```
fab
```

```
$ fab dev run  # for release server, `fab release run`
```

githubsshfabfile。

Heroku Django。

HerokuDjangoHeroku Django

```
django-admin.py startproject --template=https://github.com/heroku/heroku-django-
template/archive/master.zip --name=Procfile YourProjectName
```

GunicornWhiteNoiseDjango。Heroku

Heroku

```
git init
git add -A
git commit -m "Initial commit"

heroku create
git push heroku master

heroku run python manage.py migrate
```

Django。 LinuxNginx + Gunicorn + SupervisorUbuntu

。

1. nginx - HTTP;
2. gunicorn - 'Green Unicorn'UNIXPython WSGI HTTP;
3. supervisor - /UNIX。django / celery / celery cam;

`/home/root/app/src/root。 /home/root/app/env/` path。

NGINX

nginx。 nginx`sudo apt-get install nginx` 。 nginx`/etc/nginx/sites-enabled/yourapp.conf`。`default.conf` - 。

bellownginx conf;gunicorn。nginxgunicorn。 。

```
# your application name; can be whatever you want
upstream yourappname {
    server        unix:/home/root/app/src/gunicorn.sock fail_timeout=0;
}

server {
    # root folder of your application
    root        /home/root/app/src/;

    listen        80;
    # server name, your main domain, all subdomains and specific subdomains
    server_name   yourdomain.com *.yourdomain.com somesubdomain.yourdomain.com

    charset       utf-8;

    client_max_body_size                        100m;
```

```
    # place where logs will be stored;
    # folder and files have to be already located there, nginx will not create
    access_log        /home/root/app/src/logs/nginx-access.log;
    error_log         /home/root/app/src/logs/nginx-error.log;

    # this is where your app is served (gunicorn upstream above)
    location / {
        uwsgi_pass  yourappname;
        include     uwsgi_params;
    }

    # static files folder, I assume they will be used
    location /static/ {
        alias         /home/root/app/src/static/;
    }

    # media files folder
    location /media/ {
        alias         /home/root/app/src/media/;
    }

}
```

GUNICORN

GUNICORNdjango◦ `pip install gunicornpip install gunicorn` ◦

```
#!/bin/bash

ME="root"
DJANGODIR=/home/root/app/src # django app dir
SOCKFILE=/home/root/app/src/gunicorn.sock # your sock file - do not create it manually
USER=root
GROUP=webapps
NUM_WORKERS=3
DJANGO_SETTINGS_MODULE=yourapp.yoursettings
DJANGO_WSGI_MODULE=yourapp.wsgi
echo "Starting $NAME as `whoami`"

# Activate the virtual environment
cd $DJANGODIR

source /home/root/app/env/bin/activate
export DJANGO_SETTINGS_MODULE=$DJANGO_SETTINGS_MODULE
export PYTHONPATH=$DJANGODIR:$PYTHONPATH

# Create the run directory if it doesn't exist
RUNDIR=$(dirname $SOCKFILE)
test -d $RUNDIR || mkdir -p $RUNDIR

# Start your Django Gunicorn
# Programs meant to be run under supervisor should not daemonize themselves (do not use --
daemon)
exec /home/root/app/env/bin/gunicorn ${DJANGO_WSGI_MODULE}:application \
  --name root \
  --workers $NUM_WORKERS \
  --user=$USER --group=$GROUP \
```

```
  --bind=unix:$SOCKFILE \
  --log-level=debug \
  --log-file=-
```

gunicorn

```
sudo chmod u+x /home/root/app/src/gunicorn_start
```

`./gunicorn_start`gunicorn

SUPERVISOR

◦ `sudo apt-get install supervisor` ◦

◦ `/etc/supervisor/conf.d/your_conf_file.conf.conf`

```
[program:yourappname]
command = /home/root/app/src/gunicorn_start
user = root
stdout_logfile = /home/root/app/src/logs/gunicorn_supervisor.log
redirect_stderr = true
```

`[program:youappname]`◦ `stdout_logfile`◦

◦ Ubuntu◦

```
Ubuntu version 14.04 or lesser
```

`sudo supervisorctl reread` **- > yourappnameavailable**

`sudo supervisorctl update` **- >**`sudo supervisorctl update`;**yourappname**

```
Ubuntu 16.04
```

```
sudo service supervisor restart
```

```
sudo supervisorctl status yourappname
```

```
yourappname RUNNING pid 18020, uptime 0:00:50
```

◦

apache / nginx

Apache / Nginx◦ `DEBUG`◦ Apache / Nginx

```
python manage.py runserver --insecure
```

LAN`staticfiles`◦`INSTALLED_APPS`◦

https://riptutorial.com/zh-TW/django/topic/2792/

53:

Examples

>>/

requirementsdeployment toolsDjango。 DjangoScoops。

```
repository/
    docs/
    .gitignore
    project/
        apps/
            blog/
                migrations/
                static/ #( optional )
                    blog/
                        some.css
                templates/ #( optional )
                    blog/
                        some.html
                models.py
                tests.py
                admin.py
                apps.py #( django 1.9 and later )
                views.py
            accounts/
                #... ( same as blog )
            search/
                #... ( same as blog )
        conf/
            settings/
                local.py
                development.py
                production.py
            wsgi
            urls.py
        static/
        templates/
    deploy/
        fabfile.py
    requirements/
        base.txt
        local.txt
    README
    AUTHORS
    LICENSE
```

appsconfuser created applicationscore configuration folder 。

projectstatictemplateshtml markup。

blog accountssearchstatictemplates。

django

statictemplatesapp ex. blogex. blog/blog/base.html/base.html。

blogsearchtemplatesbase.htmlviews。

```
(Project Structure)
.../project/
    apps/
        blog/
            templates/
                base.html
        search/
            templates/
                base.html

(blog/views.py)
def some_func(request):
    return render(request, "/base.html")

(search/views.py)
def some_func(request):
    return render(request, "/base.html")

## After creating a folder inside /blog/templates/(blog) ##

(Project Structure)
.../project/
    apps/
        blog/
            templates/
                blog/
                    base.html
        search/
            templates/
                search/
                    base.html

(blog/views.py)
def some_func(request):
    return render(request, "/blog/base.html")

(search/views.py)
def some_func(request):
    return render(request, "/search/base.html")
```

https://riptutorial.com/zh-TW/django/topic/4299/

S. No		Contributors
1	Django	A. Raza, Abhishek Jain, Aidas Bendoraitis, Alexander Tyapkov, Ankur Gupta, Anthony Pham, Antoine Pinsard, arifin4web, Community, e4c5, elbear, ericdwang, ettanany, Franck Dernoncourt, greatwolf, ilse2005, Ivan Semochkin, J F, Jared Hooper, John, John Moutafis, JRodDynamite, Kid Binary, knbk, Louis, Luis Alberto Santana, lxer, maciek, McAbra, MiniGunnR, mnoronha, Nathan Osman, naveen.panwar, nhydock, Nikita Davidenko, noɥʎdʎzɐɔ, Rahul Gupta, rajarshig, Ron, ruddra, sarvajeetsuman, shacker, ssice, Stryker, techydesigner, The Brewmaster, Thereissoupinmyfly, Tom, WesleyJohnson, Zags
2	ArrayField - PostgreSQL	Antoine Pinsard, e4c5, noɥʎdʎzɐɔ
3	Django Rest Framework	The Brewmaster
4	DjangoCRUD	aisflat439, George H.
5	Django	Aidas Bendoraitis, aisflat439, Carlos Rojas, Ivan Semochkin, Rexford, Simplans
6	Django	4444, Ahmed Atalla
7	F	Antoine Pinsard, John Moutafis, Linville, Omar Shehata, RamenChef, Roald Nefs
8	JSONField - PostgreSQL	Antoine Pinsard, Daniil Ryzhkov, Matthew Schinckel, noɥʎdʎzɐɔ, Omar Shehata, techydesigner
9	Meta	Antoine Pinsard
10	RangeFields - PostgreSQL	Antoine Pinsard, noɥʎdʎzɐɔ
11	URL	knbk
12		Antoine Pinsard, Brian Artschwager, Dan Russell, Daniil Ryzhkov, fredley
13		AlvaroAV, Antoine Pinsard, George H., knbk, lxer, nhydock, Omar Shehata, Peter Mortensen, Trivial, William Reed
14	HStoreField -	noɥʎdʎzɐɔ

	PostgreSQL	
15	RedisDjango -	Majid, The Brewmaster
16	Django。	e4c5, OliPro007
17		Antoine Pinsard, e4c5, Hetdev, John Moutafis, Majid, nhydock, Rexford
18		Adrian17, Antoine Pinsard, e4c5, Kim, Matthew Schinckel, Maxime Lorant, Patrik Stenmark, SandroM, sudshekhar, Zags
19		Antoine Pinsard, dmvrtx
20		Antoine Pinsard, Antwane, coffee-grinder, e4c5, gkr, knbk, maciek, masnun, Maxime Lorant, nicorellius, pleasedontbelong, Pureferret
21		Antoine Pinsard, e4c5, knbk, Kostronor
22	CookiecutterDjango	Atul Mishra, noɥʎdʎzɐɹO, OliPro007, RamenChef
23	django	Cristus Cleetus
24		Antoine Pinsard, knbk
25		Aidas Bendoraitis, Antoine Pinsard, Daniel Rucci, ettanany, George H., knbk, NBajanca, nicorellius, RamenChef, rumman0786, sudshekhar, trpt4him
26		Antoine Pinsard, Jon Clements, mnoronha, Raito, Rexford, rigdonmr, Rishabh Agrahari, Roald Nefs, techydesigner, The_Cthulhu_Kid
27		Ian Clark
28		Ahmad Anwar, Antoine Pinsard, Evans Murithi, Kid Binary, knbk , lxer, Majid, Peter Mortensen
29		fredley, knbk
30		William Reed
31		ettanany, HorsePunchKid, John Moutafis
32		Antoine Pinsard, Brian Artschwager, Chalist, coffee-grinder, DataSwede, e4c5, Evans Murithi, George H., John Moutafis, Justin, knbk, Louis Barranqueiro, Maxime Lorant, MicroPyramid, nima, ravigadila, Sanyam Khurana, The Brewmaster
33		Aidas Bendoraitis, Alireza Aghamohammadi, alonisser, Antoine

		Pinsard, aquasan, Arpit Solanki, atomh33ls, coffee-grinder, DataSwede, ettanany, Gahan, George H., gkr, Ivan Semochkin, Jamie Cockburn, Joey Wilhelm, kcrk, knbk, Linville, lxer, maazza, Matt Seymour, MuYi, Navid777, nhydock, noɥʎdʎzɐɔ, pbaranay, PhoebeB, Rana Ahmed, Saksow, Sanyam Khurana, scriptmonster, Selcuk, SpiXel, sudshekhar, techydesigner, The_Cthulhu_Kid, Utsav T, waterproof, zurfyx
34		Burhan Khalid, Husain Basrawala, knbk, Matt Seymour, Rod Xavier, scriptmonster, techydesigner, The_Cthulhu_Kid
35		Ian Clark, John Moutafis, ravigadila
36		Adam Starrh, Alasdair, Aniket, Antoine Pinsard, Brian H., coffee-grinder, doctorsherlock, fredley, George H., gkr, lxer, Stephen Leppik, Zags
37		Antoine Pinsard, irakli khitarishvili, knbk, Medorator, naveen.panwar, The_Cthulhu_Kid
38	Celery	iankit, Mevin Babu
39		Antoine Pinsard, coffee-grinder, George H., Ivan Semochkin, noɥʎdʎzɐɔ, ssice
40		Antoine Pinsard, aquasan, Brian Artschwager, HorsePunchKid, Ivan Semochkin, John Moutafis, knbk, lxer, MarZab, Nikolay Fominyh, pbaranay, ptim, Rana Ahmed, techydesigner, Zags
41		abidibo, knbk, sudshekhar, Trivial
42	Jenkins	pnovotnak
43		RéÑjïth, sebb
44		Antoine Pinsard, ettanany
45		naveen.panwar
46		Antwane, Brian Artschwager, RamenChef
47		allo, Antoine Pinsard, Brian Artschwager, fredley, J F, knbk, Louis, Louis Barranqueiro, lxer, Maxime Lorant, NBajanca, Nils Werner, ProfSmiles, RamenChef, Sanyam Khurana, Sayse, Selcuk, SpiXel, ssice, sudshekhar, Tema, The Brewmaster
48		Antoine Pinsard, Ashutosh, e4c5, Kid Binary, knbk, Sayse, Udi
49		knbk, Rahul Gupta

50		nikolas-berlin
51		Antoine Pinsard, engineercoding, Joey Wilhelm, knbk, MicroPyramid, ravigadila, Roald Nefs
52		Antoine Pinsard, Arpit Solanki, CodeFanatic23, I Am Batman, Ivan Semochkin, knbk, lxer, Maxime S., MaxLunar, Meska, no ɥʎdʎzɐɹƆ, rajarshig, Rishabh Agrahari, Roald Nefs, Rohini Choudhary, sebb
53		Antoine Pinsard, naveen.panwar, nicorellius