Django Python web framework

CONCEPTS

manage.py is django-admin pinned to your project's settings; inside a project use manage.py.

models.py edit models
↓ `makemigrations`
Migration file in migrations/
↓ `migrate`
Database schema updated
↓ `runserver`
Dev server localhost:8000

A project holds settings and URLs; it is made of reusable apps, each with its own models, views and migrations.

PROJECT

`django-admin startproject` *<name>*
Create a project

`python manage.py startapp` *<app>*
Create an app in the project

`python manage.py check`
Run system checks

`python manage.py check --deploy`
Check production settings

`django-admin --version`
Print the Django version

Generated layout

Path	Purpose
`manage.py`	project command runner
`<name>/settings.py`	configuration
`<name>/urls.py`	root URL routes
`<app>/models.py`	database models
`<app>/views.py`	request handlers
`<app>/migrations/`	schema history

RUN

`python manage.py runserver`
Start the dev server

`python manage.py runserver` *<port>*
Use another port

`python manage.py runserver 0.0.0.0:8000`
Listen on all interfaces

RUN (cont.)

`python manage.py shell`
Python shell with Django set up

`python manage.py dbshell`
Open the database client

`python manage.py createsuperuser`
Create an admin user

`python manage.py changepassword` *<user>*
Change a user's password

MIGRATIONS

`python manage.py makemigrations`
Create migrations from model changes

`python manage.py migrate`
Apply pending migrations

`python manage.py makemigrations` *<app>*
Limit to one app

`python manage.py showmigrations`
List migrations and applied state

`python manage.py makemigrations --dry-run`
Preview without writing files

`python manage.py makemigrations --check`
Fail if migrations are missing

`python manage.py migrate` *<app> <migration>*
Migrate to a given point

`python manage.py sqlmigrate` *<app> <migration>*
Print a migration's SQL

TESTS

`python manage.py test`
Run all tests

`python manage.py test` *<app>.tests.<Class>.<method>*
Run one test

`python manage.py test --keepdb`
Reuse the test database

TESTS (cont.)

`python manage.py test --parallel`
Run tests in parallel

`python manage.py test --failfast`
Stop at first failure

`python manage.py test -k` *<pattern>*
Run tests matching a name

DATA

`python manage.py dumpdata` *<app>.<Model>* **`--indent 2`**
Export data as JSON

`python manage.py loaddata` *<fixture>*
Import a fixture

`python manage.py flush`
Empty all tables, keep migrations

STATIC & I18N

`python manage.py collectstatic`
Gather static files for deployment

`python manage.py collectstatic --noinput`
Run without prompts

SHELL QUERIES

`Model.objects.all()`
All rows

`Model.objects.filter(`*`<field>`***`=`***`<value>`***`)`**
Rows matching a condition

`Model.objects.get(pk=`*`<id>`***`)`**
One row; raises if missing or many

`Model.objects.create(`*`<field>`***`=`***`<value>`***`)`**
Insert a row

RECIPES

Change a model

```
# edit <app>/models.py
python manage.py makemigrations
python manage.py sqlmigrate <app>
0002
python manage.py migrate
```