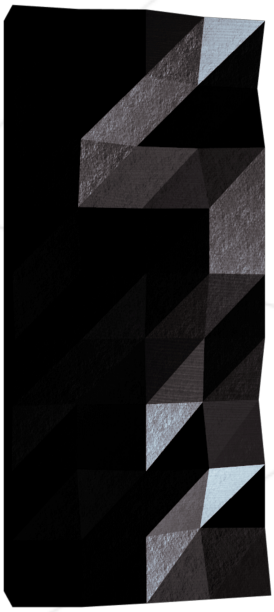

D05 - Django-Python Training

Django - ORM

Damien Cojan dcojan@student.42.fr
42 Sta ff pedago@staff.42.fr

Summary: Today we will discover the ORM Django.

Contents

Chapter I

Preamble

Chapter II

Instructions

- Only this page serve as a reference: Do not fi rm the hallway noise.
- The subject can change up to an hour before rendering.
- If no contrary information is explicitly present, you should assume that the versions of languages and frameworks used are (or later):
  - Python 3
  - HTML5
  - CSS 3
  - Javascript EM6
  - Django 1.9
  - psycopg2 2.6
- Unless otherwise indicated in the subject, the fi les in python each year on
  Python one (d01, d02 and d03) must have their end block if __name__ == '__main__': in order to insert the entry point in the case of a program, or tests in the case of a module.
- Unless otherwise indicated in the subject, each year days on
  Django will have its own application in the project to make for reasons peda- gogiques.
- The exercises are precisely ordered from simple to more complex. In no event shall we bear no attention or take into account a complex exercise if a simpler exercise is not very successful.
- Attention to the rights of your fi les and your directories.
- You must follow the rendering process for all your exercise: only the present work on your GIT repository will be evaluated in defense.
- Your exercises will be evaluated by your pool mates.
- You should leave in your repertoire no other fi le than those explicitly speci fi ed by the forward drills.
- Unless otherwise specified in the topic you should not include in your rendering:

- The files __ pycache__.
- Any migration.
  Warning, you are still advised to make the fi le migration / __ init__.py,
  it is not necessary, but simplifies the construction of migration. Do not add this fi le will not invalidate your rendering but you *must* be able to manage migration for correction in this case.
- The file created by the command collectstatic of manage.py (with the way the value of the variable STATIC_ROOT).
- The fi le Python bytecode (fi les with extension. pyc).
- database fi les (especially with sqlite).
- Any fi le or folder must or can be created by the normal behavior of the rendering work.

It is recommended that you modify your. gitignore in order to avoid accidents.

- When you need to get an accurate output in your programs, it is of course forbidden to a ffi expensive output precalculated instead of performing the exercise cor- rectly.
- You have a question ? Ask your neighbor to the right. Otherwise, try your left neighbor.
- Your reference manual called Google / man / Internet / ....
- Think discuss on the forum of your pool Intra!
- Read the examples carefully. They might require things that are not otherwise specified in the subject ...
- Please, by Thor and Odin! Re fl échissez dammit!

chapter III

Specific Rules of the day

- The interpreter is to use python3.
- Each exercise is independent. If among the features requested, some have already been made in previous years, the dupliquez- in the exercise LYING rant.
- You must work in a database postgresql named formationdjango
  and create a named role djangouser, whose password is " secret", which will have all the rights on it.
- Your file must be made a Django project. The project name must be that of the current day.
- We will use the concept of applying Django to separate exercises: Each exercise of the day must be in a separate Django application with the name of the corresponding year and being at the root of the rendering file.
- The Django project must be configured correctly in order to qualify for the exercises. No change configurations will be permitted in defense.
- You shall make no migration with your work.
- In each year whose mention cartridge ORM, you must use the ORM Django. No line SQL should be written.
- In each year whose mention cartridge SQL, you must use the li- brairie psycopg2 and e ff ectuer all requests SQL.

An example of typical structure for rendering the krichard student, on d42 day and including two years:

```
| - krichard |
        | -. |
        | - .. |
        | - .git |
        | - .gitignore |
        | - d42 |
        | | - __init__.py |
        | | - settings.py |
        | | - urls.py |
        | | - wsgi.py |
        | - ex00 |
        | | - admin.py |
        | | - apps.py |
        | | - forms.py |
        | | - __init__.py |
        | | - models.py |
        | | - tests.py |
        | | - urls.py |
        | | - views.py |
        | - ex01 |
        | | - admin.py |
        | | - apps.py |
        | | - forms.py |
        | | - __init__.py |
        | | - models.py |
        | | - tests.py |
        | | - urls.py |
        | | - views.py |
        | - manage.py
```

.

Be smart: refactor your code and make it easier to reuse, you can save time.

Chapter IV

Exercise 00

	Exercise: 00
00 Exercise: SQL - construction of a table	
Render Folder: *ex* 00 / Files to	
make: Functions Authorized:	
Notes: n / A	

Creez a named Django application ex00 and views to the inside of it that should be accessible via the URL: 127.0.0.1:8000/ex00/in

This view must create a SQL table in Postgresql using the library psycopg2 and return a page with "OK" on success, or otherwise an error message describing the problem.

The SQL table must meet this description. :

- It must be named ex00_movies.
- It should only be created if it does not already exist.
- It must contain only the following fields:
  - title: chain variable charactere, unique, with a maximum size of 64 bytes, not zero.
  - episode_nb: integer, PRIMARY KEY.
  - opening_crawl: text may be zero, no size limit.
  - director: string variable characters, not zero, with a maximum size of 32 bytes.
  - producer: chain variable character, not zero, with a maximum size of 128 bytes.
  - release_date: Date (no time), not zero.

Chapter V

Exercise 01

	Exercise: 01
01 Activity: ORM - building a table	
Render Folder: *ex* 01 / files to	
make: Functions Permitted	
Remarks: n / A	

Creez an application named ex01, and therein a model Django appointed movies having exactly these fields:

- **title: character string, single, with a maximum size of 64 bytes.**
- **episode_nb: integer, PRIMARY KEY.**
- **opening_crawl: text - can be null.**
- **director: string with a maximum size of 32 bytes, not zero.**
- **producer: string with a maximum size of 128 bytes, not zero.**
- **release_date: Date (no time), not zero. This model must also redefine the method __ str__ in order that it returns the at- tribute title**

Chapter VI

Exercise 02

	Exercise: 02
Exercise 02: SQL - Data insertion	
Render Folder: *ex* 02 / Files to	
make: Functions Authorized:	
Notes: n / A	

You must create a named Django application EX02, and therein, the views ac- transferable via the following URL:

- **127.0.0.1:8000/ex02/init: must create a table to be in every respect the same characteristics as that required for ex00 the diff erence except that it will be called ex02_movies.**

  It should return a page ffi singing "OK" on success, or otherwise an error message describing the problem.

- **127.0.0.1:8000/ex02/populate: must be inserted into the table created by the pre cedente view the following** data:

  - episode_nb 1 - title: The Phantom Menace - director George Lucas - producer Rick McCallum - release_date: 1999-05-19
  - episode_nb: 2 - title: Attack of the Clones - director George Lucas - produ- cer: Rick McCallum - release_date: 2002-05-16
  - episode_nb: 3 - title: Revenge of the Sith - director George Lucas - producer Rick McCallum - release_date: 2005-05-19
  - episode_nb: 4 - title: A New Hope - director George Lucas - producer Gary Kurtz Rick McCallum - release_date: 1977-05-25
  - episode_nb: 5 - title: The Empire Strikes Back - director Irvin Kershner - producer: Gary Kutz, Rick McCallum - release_date: 1980-05-17

- episode_nb: 6 - title: Return of the Jedi - director Richard Marquand - pro- ducer Howard G. Kazanjian George Lucas, Rick McCallum - release_date: 1983-05-25

- episode_nb: 7 - title: The Force Awakens - director JJ Abrams - producer Kathleen Kennedy, JJ Abrams, Bryan Burk - release_date: 2015-12-11 It should return a page ffi song for each insertion "OK" if successful, or otherwise: a message describing the name of the film, followed by the problem.

- 127.0.0.1:8000/ex02/display: is a ffi expensive all the data in the table ex02_movies in a table HTML including any null fields.

  If no data is available, or if an error occurs, the page is simply a ffi expensive " No data available. "

Chapter VII

Exercise 03

	Exercise: 03
Exercise 03: ORM - Data insertion	
Render Folder: *ex* 03 / Files to	
make: Functions Authorized:	
Notes: n / A	

Creez a new app named Django EX03 and create in it a Django model similar in all respects to that of the ex01.

This app should contain the views accessed via the following URLs:

- 127.0.0.1:8000/ex03/populate: must include in the table created by the pre cedente view the following data:

  - episode_nb 1 - title: The Phantom Menace - director George Lucas - producer Rick McCallum - release_date: 1999-05-19
  - episode_nb: 2 - title: Attack of the Clones - director George Lucas - produ- cer: Rick McCallum - release_date: 2002-05-16
  - episode_nb: 3 - title: Revenge of the Sith - director George Lucas - producer Rick McCallum - release_date: 2005-05-19
  - episode_nb: 4 - title: A New Hope - director George Lucas - producer Gary Kurtz Rick McCallum - release_date: 1977-05-25
  - episode_nb: 5 - title: The Empire Strikes Back - director Irvin Kershner - producer: Gary Kutz, Rick McCallum - release_date: 1980-05-17
  - episode_nb: 6 - title: Return of the Jedi - director Richard Marquand - pro- ducer Howard G. Kazanjian George Lucas, Rick McCallum - release_date: 1983-05-25

- episode_nb: 7 - title: The Force Awakens - director JJ Abrams - producer Kathleen Kennedy, JJ Abrams, Bryan Burk - release_date: 2015-12-11 It should return a page ffi song for each insertion "OK" if successful, or otherwise: a message describing the problem.

- **127.0.0.1:8000/ex03/display: must return a page HTML a ffi singing the integrator ity of the data in the table movies, formatted in a table HTML,**
  including any null fields.

  If no data is available, the page is simply a ffi expensive " No data available. "

In defense, migration will be before the tests.

Chapter VIII

Exercise 04

	Exercise: 04
Exercise 04: SQL - data removal	
Render Folder: *ex* 04 / Files to	
make: Functions Authorized:	
Notes: n / A	

Creez an app named ex04. It must contain the views accessed via urls sui- lowing:

- 127.0.0.1:8000/ex04/init: must create a table to be in every respect the same characteristics as that required for the app the ex00 the diff erence except that it will be called ex04_movies.

  It should return a page ffi singing "OK" on success, or otherwise an error message describing the problem.

- 127.0.0.1:8000/ex04/populate: must include in the table created by the pre cedente view, the data described in exercise EX02.

  This view must always reinsert any data that was deleted. It should return a page ffi song for each insertion "OK" on success, or otherwise: a message describing the problem.

- 127.0.0.1:8000/ex04/display: is a ffi expensive all the data contained in the table ex04_movies in a table HTML, including any null fields.

  If no data is available, or if the error page is simply a ffi expensive " No data available. "

- 127.0.0.1:8000/ex04/remove: is a ffi expensive form HTML containing a combo films titles and a button submit appointed remove.

The films securities are those contained in the table ex04_movies.

When the form is validated, the fi lm selected is deleted from the database and the form is rea ffi ket with the list of remaining fi setting á day films. If no data is available or if an error occurs, the page is simply a ffi expensive " No data available. "

Chapter IX

Exercise 05

	Exercise: 05
Exercise 05: ORM - deleting data	
Render Folder: *ex* 05 / Files to	
make: Functions Authorized:	
Notes: n / A	

Creez a new app named Django ex05 and create in it a model similar in all respects to that of ex01.

This app should contain the views accessed via the following URLs:

- 127.0.0.1:8000/ex05/populate: must include in the model created for this application, the data described in exercise EX03.

  This view must always reinsert any data that was deleted. It should return a page ffi song for each insertion "OK" on success, or otherwise: a message describing the problem.

- 127.0.0.1:8000/ex05/display: is a ffi expensive all the data in the table movies This app in a table HTML; including any null fields.

  If no data is available, or if an error occurs, the page is simply a ffi expensive " No data available. "

- 127.0.0.1:8000/ex05/remove: is a ffi expensive form HTML containing a combo films titles and a button submit appointed remove.

  The films are those securities in the model movies this application. When the form is validated, the fi lm selected is deleted from the database and the form is rea ffi ket with the list of remaining fi setting á day films.

If no data is available, the page is simply a ffi expensive " No data available. "

In defense, migration will be before the tests.

Chapter X

Exercise 06

	Exercise: 06
Exercise 06: SQL - Update a given	
Render Folder: *ex* 06 / Files to	
make: Functions Authorized:	
Notes: n / A	

Creez a new app named Django EX06. It must contain the views accessed via the following URLs:

- 127.0.0.1:8000/ex06/init: must create a table that will be exactly the same as FY00 at di ff erence except that it will be called ex06_movies and it will have the following additional fields:
  - created datetime (date and time), which, á creation, must be automatically assigned á the current date and time.
  - updated datetime (date and time), which, á creation, must be automatically assigned á the current date and time and automatically update each update with the following trigger:

```
CREATE OR REPLACE FUNCTION update_changetimestamp_column () RETURNS TRIGGER
AS $$ BEGIN

NEW.updated = now (); NEW.created =
OLD.created; RETURN NEW; END;


$$ language 'plpgsql';
CREATE TRIGGER BEFORE UPDATE ON update_films_changetimestamp ex06_movies FOR
EACH ROW EXECUTE PROCEDURE update_changetimestamp_column ();
```

- 127.0.0.1:8000/ex06/populate: must populate the table created by the preceding dente view with the data described in exercise EX02.

  It should return a page ffi song for each insertion "OK" if successful,

or otherwise: a message describing the problem.

- 127.0.0.1:8000/ex06/display: is a ffi expensive all the data in the table ex06_movies in a table HTML.

  If no data is available or if an error occurs, the page is simply a ffi expensive " No data available. "

- 127.0.0.1:8000/ex06/update: to manage the sending and receipt of a lar formulated. This should allow to choose a film from a dropdown list con- taining the titles of the films of the table ex06_movies, and write text in a deuxème field. By validating the form, the view must replace in the table

  ex06_movies, field crawling_text the film selected by the text entered in the form.

  If no data is available or if an error occurs, the page is simply a ffi expensive " No data available. "

Chapter XI

Exercise 07

	Exercise: 07
Exercise 07: ORM - Update a given	
Render Folder: *ex* 07 / Files to	
make: Functions Authorized:	
Notes: n / A	

Creez a new app named Django EX07 and create in it a model similar in all respects to that of the ex01, the di ff erence except that you must add the following additional fields:

- created datetime (date and time), which, á creation, must be automatically assigned á the current date and time.
- updated datetime (date and time), which, á creation, must be automatically assigned á the current date and time and automatically update with each update.

This app should contain the views accessed via the following URLs:

- 127.0.0.1:8000/ex07/populate: people model previously created with the same data as the EX02

  It should return a page ffi song for each insertion "OK" on success, or otherwise: a message describing the problem encountered.

- 127.0.0.1:8000/ex07/display: a ffi che all the data in the table movies in a table HTML.

  If no data is available or if an error occurs, the page is simply a ffi expensive " No data available. "

- 127.0.0.1:8000/ex07/update: must manage the sending and receipt of a form. This should allow to choose a film from a dropdown list containing

the titles of the films contained in the model movies this application, and write text in a deuxème fields.

By validating the form, the view must modify the model movies of this application, the fields crawling_text the film selected with the text entered in the form.

If no data is available or if an error occurs, the page is simply a ffi expensive " No data available. "

In defense, migration will be before the tests.

Chapter XII

Exercise 08

	Exercise 08 Exercise 08: SQL -
Foreign Key	
Render Folder: *ex* 08 / Files to	
make: Functions Authorized:	
Notes: n / A	

Creez a new app named Django ex08. This app should contain the views sible accessible via the following URLs:

- 127.0.0.1:8000/ex08/init: Must create two tables. The first must be named ex08_planets and have the following fields:
    - id: serial, primary key
    - name: chain variable caratères, unique, non-null, a maximum size of 64.
    - climate: chain variable caratères.
    - diameter: whole.
    - orbital_period: whole.
    - population: whole big.
    - rotation_period: whole.
    - surface_water: real.
    - ground : caratères string variable, with a maximum size of 128. The second should be named ex08_people and have the following fields:
    - id: serial, primary key.
    - name: chain variable caratères with a maximum size of 64.

- Birth_Year: chain variable caratères, maximum size of 32.
- gender: chain variable caratères, maximum size of 32.
- eye_color: chain variable caratères, maximum size of 32.
- hair_color: chain variable caratères, maximum size of 32.
- height: whole.
- mass: real.
- homeworld: chain variable caratères, with a maximum size of 64, foreign key, referencing the column name of the table 08_planets

- 127.0.0.1:8000/ex08/populate: Must populate two tables by copying the contents fi les provided people.csv and planets.csv in the corre- sponding tables, respectively: ex08_people and ex08_planets.

  This view must return a page for each song ffi fi le "OK" on success, or otherwise: a message describing the problem.

- 127.0.0.1:8000/ex08/display: a ffi che all character names, their original nete ceiling and climate, said that climate is windy or part (' windy '), Sorted alphabetically character names. If no data is available or if an error occurs, the page is simply a ffi expensive " No data available. "

Find out about the method of copy_from psycopg2

Chapter XIII

Exercise 09

	Exercise 09 Exercise 09: ORM -
Foreign Key	
Render Folder: *ex* 09 / Files to	
make: Functions Authorized:	
Notes: n / A	

Creez a new app named Django ex09 and create in this two models. The first model that you create must be named Planets and must contain the following fields:

- name: string, unique, non-null ,. maximum of 64
- climate: string of characters.
- diameter: whole.
- orbital_period: whole.
- population: whole big.
- rotation_period: whole.
- surface_water: real.
- ground : string of characters.

This model must also redefine the method __ str __ () in order that it returns the attribute name.

The second model that you create must be named People and must contain the following fields:

- name: string, max 64.

- Birth_Year: string, max 32.
- gender: string, max 32.
- eye_color: string, max 32.
- hair_color: string, max 32.
- height: whole.
- mass: real.
- homeworld: string, max 64, foreign key referencing the column name
  of the table Planets this application. This model must also redefine the method __ str __ () in order that it

returns the attribute name.

Also create this app, a view that must be accessible at
127.0.0.1:8000/ex09/display.
This view is a ffi expensive in a table HTML All character names, their home planet and the climate, which said climate is windy or part (' windy '), Sorted by alphabetical order of the names of characters.

If no data is available, the view is a ffi expensive text " No data available, please Use the following command line before use " followed by a com- mande line.

This command line should be the one to run from the root of your made in order to fit into the previously created templates, all data in the fi le ex09_initial_data.json (provided in the resources of the day).

You will need to provide this fi le with your rendering.

In defense, migration will be before the tests.

Chapter XIV

Exercise 10

	Exercise: 10
Exercise 10: ORM - Many to Many	
Render Folder: *ex* 10 / Files to make: file_to_turn_in	
Permitted functions: functions_authorized	
Remarks: n / A	

Creez a new app named Django ex10 and create in it three models:

- Planets and People: Both models should be identical in all respects to those of the year ex09.

- movies: This model must be identical in all respects to that of ex01 the di ff erence except that you must add the field characters.
  Field type " many to many " with model People and it will list all the characters in a film and are in the table

  People.

The fi xtures required for settlement patterns are present in the fi le ex10_initial_data.json included among the resources of the day.

You must also create this app accessible via the URL view 127.0.0.1:8000/ex10.
This is a ffi expensive a form with these fields (chacuns required):

- Movies minimum release dates: dated
- Movies maximum release dates: dated
- Planet diameter Greater Than: number
- Character Gender: drop-down list containing the di ff erent values available in the field gender People of model. It should not be there several times the same value.

Once validated, the view should look for, and return a ffi first price or results. A result is a character whose gender matches the field 'Character gender' with a film in which it is present and whose output date falls between Movies minimum release dates and Movies maximum release date with its ori- gin planet whose diameter is greater than or equal to Planet diameter Greater Than.

If there are no results matching the query " Nothing Corresponding to your research " must ffi was expensive.

Each result should ffi was expensive on a line with these items (not necessarily in this order):

- The character's name
- His genre
- The title of the film
- The name of his home planet (homeworld)
- The diameter of its home planet

For example: the results of feminine characters, whose films were released between '1900-01-01' and '2000-01-01' and whose home planet has a diameter SUPREME laughing 11000 are:

- A New Hope - Leia Organa - female - Alderaan - 12500
- The Phantom Menace - Padmé Amidala - female - Naboo - 12120
- Return of the Jedi - Leia Organa - female - Alderaan - 12500
- Return of the Jedi - Mothma - female - Chandrila - 13500
- The Empire Strikes Back -Leia Organa - female - Alderaan - 12500

Several characters can be in the same film, and the same character may appear in several films. It's called a relationship many to many (many to many). In this case, an intermediate table must be created between the two tables. Each row of this table is through an association (one) of two numbers: the first row referencing a table of films, the second row referencing a table of characters (or in reverse order). Once you have built your models and performed migration, you will see this table via postgre console.