



# D07 - Django-Python Training

## Django - Advanced

Damien Cojan [dcojan@student.42.fr](mailto:dcojan@student.42.fr)  
42 Staff [pedago@staff.42.fr](mailto:pedago@staff.42.fr)

*Summary: Today we'll find some advanced features  
Django.*

# Contents

<b>I</b>	<b>Preamble</b>	<b>2</b>
<b>II</b>	<b>instructions</b>	<b>5</b>
<b>III</b>	<b>Specific Rules of the day</b>	<b>7</b>
<b>IV</b>	<b>Exercise 00</b>	<b>8</b>
<b>V</b>	<b>FY01</b>	<b>10</b>
<b>VI</b>	<b>exercise 02</b>	<b>12</b>
<b>VII</b>	<b>exercise 03</b>	<b>14</b>
<b>VIII</b>	<b>Exercise 04</b>	<b>16</b>
<b>Exercise IX</b>	<b>05</b>	<b>17</b>
<b>X</b>	<b>exercise 06</b>	<b>18</b>

# Chapter I

## Preamble

Happiness frameworks

I wanted to build a small shelf to store condiments. Having done some carpentry before, I had a good idea of what I needed: some wood and some basic tools. A meter, a saw, a level and a hammer.

Besides, if I wanted to build a house, I need it also. So I went to a hardware store and I asked the seller where I could find a hammer.

"- A hammer?" He replied. "Nobody buys hammers nowadays you know. They are a bit old-fashioned. "

Surprised, I asked why.

- "Well, the problem with hammers is that there are lots of different kinds. nail puller hammers, sledge hammers, hammers upholsterer. . . What would happen if you were buying a type of hammer and realize that you need another type later? You should buy another hammer to your next task. It turns out that most people really want a single hammer that can be used for the majority of tasks they may face in their lives. "

- "It seems logical to me. Can you tell me where I can find a universal salt hammer? "

- "No, we do not sell them anymore. They are obsolete. "

- "Really ? I thought you just said that universal hammer was fine. "

- "It turns out that if you make a single hammer that can be used for all kinds of tasks, it is not really good at any of them. Hammer a nail with a

mass is not very efficient. And to kill your girlfriend, nothing beats an upholsterer's hammer. "

- "It's clear ! So if nobody buys Universal Hammers, and you sell more hammers old, what hammers do you sell? "

- "Actually, we do not sell."

- "So. . . "

- "According to our research, what people need is not a universal hammer at all. It is always better to have the right hammer for the good job. So we started selling hammer factories, capable of producing any hammer that may interest you. All you need is to fill the factory workers start machinery, buy raw materials, pay expenses and presto, you have \* exactly \* the kind of hammer you need in a flash. "

- "But I do not want to buy a factory hammers. . . "

- "Perfect. Because we sell more. "

- "Wait, you just said that. . . "

- "We found that most people do not need a full factory hammers. Some, for example, will never need an upholsterer's hammer. (Maybe they have no ex. Or maybe they killed them with ice picks.). So there is no reason for someone to buy a hammer factory for all types of hammers. "

- "Yes it's sure."

- "So, instead, we started selling the plans for construction of the factory hammers in order that our customers can build their own factories, completely customized to produce only the types of hammers they need."

- "Let me guess. You sell them more. "

- "No. Of course. It turns out that people do not want to build a factory just to make some hammers. Let the construction of factories to factories construction experts, this is what I always say! "

- "And I agree with you on that."

- "And yes. So we stopped selling these plans and we started selling hammer factories factories. Each is built by our experts in the hammer factory factory business, in order that you do not have to worry about trivial details of the construction of a factory. Yet you

have all the benefits of having your own customized factory producing your own custom hammers, sticking to your specific designs for hammer. "

- "Uh, it does not seem to me really. . . "

- "I know what you'll say! ! . . . and we sell more besides. Apparently, few people were buying the hammer factory factories, so we found a solution to this problem. "

- "Hmm."

- "We took the time to take stock of our technical infrastructure, we determined that people were developing a frustration with having to manage and operate a hammer factory factory, as it produced the factory. This kind of additional constraint can be tedious when you find yourself in a scenario where you also use a meter manufacturing factory, a saw mill and factory levels manufactures factory. Besides a conglomerate of wood processing. We have objectively assessed the situation and determined that it was too complex for someone who just wanted to create a shelf for spices. "

- "No kidding ?"

- "So this week, we bring to market a factory building factory manufactures tools of all kinds, whereby your different tools factory factories can be created from a single mill unified . The factory factory factory will produce only factory manufactures ment réelle- you need, and so makes these factories produce a single factory based on your custom tools specifications. You will \* exactly \* the hammer you need, and exactly the right meter for your task, just by pressing a button (though you'll probably few file configuration so that everything works as you expect).

- "So you do not have hammers? Not at all ?"

- "No. If you really want a shelf high quality condiments, industrial standard, you really need something more sophisticated than a simple hammer bought at the hardware store. "

- "Okay. . . Good. It takes what it takes. If that's the way we do now, it is necessary that I put myself. "

- "Excellent! "

- "It comes with documentation, right?" Source: [SametMax](#)

# 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
  - psycpg2 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.
- The file created by the command `collectstatic` of `manage.py` ( with the way the value of the variable `STATIC_ROOT`).
- The file Python bytecode (files with extension. `.pyc`).
- database files (especially with `sqlite`).
- Any file 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 correctly.
- 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

- During this pool of day you'll develop a unique site. It aims to offer items to view. It allows connected users to publish new articles or to save it as a favorite.
- You have the freedom to name this site as you please and to focus its thematic tick to your taste (news, fan fiction, new erotic, etc ...).
- You can choose the database you want, as long as it is compatible with the native ORM Django
- Your rendering will take the form of a single project Django. He will not have the usual cutting exercises. Each rajoutent project functionality. It is this and its implementation that will be noted.
- It is forbidden to implement any url "hard". You should always refer to name the url. Whether the views or templates.
- It is forbidden to implement any view as functions. You should use generic views.
- Remember that the exercises will be corrected in order.
- The default language of your site is English. The content of the database can be anything. Latin if it suits you.
- You must leave the default management application.




Do not waste time on what is not asked!



## Chapter IV

### Exercise 00

	Exercise: 00
00 Exercise: Model building - Generic Class	
Render Folder: <i>ex 00</i> / Files to	
make: Functions Authorized:	
Notes: n / A	

Create a new project. You can name it as you please and organize the structure of the application as you please. Think a logical structure and easy to understand facilitate corrections.

Implement the following models:

- Article: Content articles and some metadata. It should contain the following fields:
  - title: Article Title. characters string with a maximum size of 64. No null.
  - author: Author of the article. Reference recording of the User model. No null.
  - created: Date and time of creation complete. Must be filled automatically with creation. No null.
  - synopsis: Article summary. characters string with a maximum size of 312. No null.
  - happy : The article itself. This is a text. No null. The `__str ()` method must be `__`

'override' to return 'title'

- **UserFavouriteArticle:** Items favorites recorded by a user. Ne must contained the following fields:
  - **user:** Reference recording of the User model. No null.
  - **Article:** Reference recording of the Article model. No null. The `__str()` method must be `__overriden` to return 'title' which is in the Article model.

Once done, we can move on to the real objective of this first exercise. Using only generic views (except 'View' you do not have permission to inherit directly), you must implement the following fonctionnali- ties into your site. Each of these features should have its own URL:

**Articles:** Page HTML a ffi song in table form HTML all fields (with the exception tion happy) of all items stored in the table Article.

The table must have a header indicating the title of each column.

**Home:** Url imposed: '127.0.0.1:8000. redirects to goods

**Login:** Page HTML a ffi singing a type of form POST. Logue non user

logged with a user name and a password. On error, the page has a ffi expensive a message describing the error. On success, the view must redirect 'Home'.


You must also provide at least five articles of examples from three utiliza- ers di ff erent. Provide fi xtures if necessary. The content of the articles does not matter, the 'lorem ipsum' is perfect.



No css formatting is required as part of this exercise.

# Chapter V

## Exercise 01

	Activity: 01 year 01: Generic
Class again	
Render Folder: ex01 / files to	
make: Functions Permitted	
Remarks: n / A	

Using only generic views (except 'View' you do not have permission to inherit directly), you must implement the following fonctionnali- ties into your site. Each of these features should have its own URL:

**Publications:** Page HTML a ffi song, in table form HTML the fields 'Title' 'synopsis' and 'Created' of all items stored in the model Article ' whose author is the currently logged on user.

For each item, you must also implement a link whose URL should contain the identifier of the said Article, leading functionality 'Detail' the ar- ticle concerned.

The table must have a 'header' indicating the title of each column.

**Detail :** Page HTML ffi song has any field of a given article being in base of data. The identi fi cation of this article must be in the url.

The field layout is free.

You must also add to each item in this table HTML functionality goods last year, a link to the 'Detail' the said Article.

**Logout:** Link déloguant a logged-in user. The place where is the link to logout does not matter in this exercise, as long as it is visible and accessible. Once logged out, the user is redirected to 'Home'

**Favorites:** page HTML a ffi song as a list of links, titles


Bookmark saved articles by the currently logged in user.

Each link, the url must contain the identifier of the item in question must carry out functionality 'Detail' the said Article.

You must provide at least one user having two favorites di ff erent.

# Chapter VI

## Exercise 02

	Exercise: 02
Exercise 02: Generic Class - CreateView	
Render Folder: ex 02 / Files to	
make: Functions Authorized:	
Notes: n / A	

Using only CreateView, you must implement the following functionalized bedded in your site. Each of these features should have its own URL:

**Register:** Page HTML containing a type of form 'POST' allowing a UTI

lisateur not logged create a new user account.

The form must request a login, a password and a confirmation password. This form must be accessible to a URL that is exclusively dedicated to him and that must end with 'Register'.

**Publish:** Page HTML containing a type of form 'POST' allowing a UTI

lisateur logged publish a new article. The 'author' must not be a ffi ket, it must be completed in the view during the validation process. You must use an object 'Form' created by your View to generate your form (not <input> Hand typed to the fields in your form!)

Add a link to this functionality in the template functionality Publications.

**Add to favorite:** Page HTML containing a type of form 'POST' is Trouville

efore in the detail page of an article. No field should be visible. Field Article ' must be pre-filled with the ID of the current article and upon validation, the field 'User' should be filled with the user ID logged. This mechanism Me- adds Article current favorites of the connected user.




No CSS formatting is required as part of this exercise.



You knew Django offers all ready forms?

# Chapter VII

## Exercise 03

	Exercise: 03
Exercise 03: Template Tags and Filters	
Render Folder: <i>ex 03</i> / Files to	
make: Functions Authorized:	
Notes: n / A	

In this exercise you have to create a menu that is visible from ALL pages of the site.

All links lead to this menu of features YOU HAVE CREATED.

If you are missing something ... ask you questions.

This menu has the following items:

- **Home:** Link functionality 'Home' ( which in turn redirects to 'Articles' you remember ?).
- **Last Articles:** Link functionality Article. You can customize the name of this link according to any theme you have chosen (but always in English!).
- If a user is not connected:
  - **Register:** Link functionality 'Register'
  - **Login:** functionality 'Login'. Here is a little different because it is not a simple link, you need to put the entire form IN the menu. It will be available on all pages, not just on the dedicated page that you have created.

It must however always a ffi expensive mistakes if form inva-

lide.

- If a user is logged:
  - **Favorites:** Link functionality 'Favorites'
  - **Publications:** Link functionality 'Publications'
  - **Logged as <user>:** Simple text indicating that the user is logged in. <User> must obviously be replaced with the name of the connected user.
  - **Logout:** functionality 'Logout'. Now you have a specific place to put this link.

Using tags and filters, changed ez template listing all the items so that:

- The synopsis is reduced to a maximum of 20 characters, beyond which the following should be replaced by three dots. You must provide an example that is demonstrating.
- The list of items to be sorted by date, from most recent to oldest.
- An additional column mentions how long the article was published.




No CSS formatting is required as part of this exercise.



## Chapter VIII


### Exercise 04

	Exercise 04 Exercise 04:
Bootstrap	
Render Folder: <i>ex 04</i> / Files to	
make: Functions Authorized:	
Notes: n / A	

Using Bootstrap, give to your CSS formatting the same menu as the one in the image provided in the resources of the day.

## Chapter IX

### Exercise 05

	Exercise 05 Exercise 05:
Internationalization	
Render Folder: <i>ex 05</i> / Files to	
make: Functions Authorized:	
Notes: n / A	

Translate all functionality goods Site and the menu (which is visible from here also) in French based on the prefix lying in the URL.

For example: If my URL is: 127.0.0.1:8000/en/articles, then all its contents will be english.

If this URL is 127.0.0.1:8000/fr/articles, then all its contents will be French.


The content of the database and the site name does not translate. By default, the site must be in

English.

Add a link to switch from one language to another on the same page.

# Chapter X

## Exercise 06

	Exercise 06 Exercise
06: Testing	
Render Folder: ex 06 / Files to	
make: Functions Authorized:	
Notes: n / A	

Using the built-in test framework Django create tests to verify that the site out the following behaviors:

- The views favorites, publications and publish and their templates are accessible only to registered users.
- It is not possible for a user logged in to access the form for creating a new user.
- A user can not put the same article twice favorites. Your tests must imperatively wear a descriptive name, indicating exactly what they veri fi ed.

Correct any errors potentiellements revealed by such tests