

Day 08: Python Django

Django: HTTP

Summary: Now your know how to design a Django project with all the features you can dream of. It's time to share it with the world!

# Contents

Ι	Preamble	2
II	Instructions	3
III	Today's specific rules	4
IV	Exercise 00	5
V	Exercise 01	7
VI	Exercise 02	8
VII	Exercise 03	9
VIII	Exercise 04	10

# Chapter I Preamble

This will be a short preamble.

Done.

#### Chapter II

#### Instructions

Unless there is an explicit contradiction, the following instructions will be valid for all days of this Python Django Piscine.

- Only this page will serve as reference; do not trust rumors.
- Watch out! This document could potentially change up to an hour before submission.
- These exercises are carefully laid out by order of difficulty from easiest to hardest. We will not take into account a successfully completed harder exercise if an easier one is not perfectly functional.
- Make sure you have the appropriate permissions on your files and directories.
- You have to follow the submission procedures for every exercise.
- Your exercises will be checked and graded by your fellow classmates.
- On top of that, your exercises will be checked and graded by a program called Moulinette. Moulinette is very meticulous and strict in its evaluation of your work. It is entirely automated and there is no way to negotiate with it. So if you want to avoid bad surprises, be as thorough as possible.
- Exercises in Shell must be executable with /bin/sh.
- You <u>cannot</u> leave <u>any</u> additional file in your directory than those specified in the subject.
- Got a question? Ask your peer on the right. Otherwise, try your peer on the left.
- Your reference guide is called Google / man / the Internet / ....
- Remember to discuss on the piscine forum of your Intra and on Slack!
- Examine the examples thoroughly. They could very well call for details that are not explicitly mentioned in the subject...

# Chapter III Today's specific rules

 $\bullet$  Your repository will be cloned during the evaluation with the following command: git clone <repo> ~/d08

You will have to make sure your configuration (server configuration especially) will work in this location.

#### Chapter IV

#### Exercise 00

	Exercise 00
	Exercise 00: A reusable application.
Turn-	in directory: $ex00/$
Files	to turn in : Your project, the sources for your application, your
requi	irements.txt file
Allow	red functions: All the Django functionalities, setuptools, pillow

Today, you'll learn how to get your projects in production!

However, you will need a project to serve to show your mastery in server science. What a coincidence! This is precisely the subject of this exercise!

Create a Django application implementing only one page. This page will feature a picture you want in the background and a form containing a field to submit an image and a text field with a title for this picture as a parameter.

A visitor will be able to upload a picture and give it a title. The group of uploaded pictures will be displayed on a page. Each picture will feature its title.

You will also create a *project* in which you will integrate an application you've just coded. You will have to design it and install it as a package with pip.

Your project will be tested with DEBUG set at True, but you *HAVE TO* turn it in with the variable set at False.

You can add some CSS, use bootstrap, cats, summon Great Old Ones, anything that suits you. But the previous instructions will have to be observed.

You should also manage each type of files (MEDIA, STATIC...) wisely. Make sure the files are managed by the development server if (and only if) the DEBUG variable is set on True.

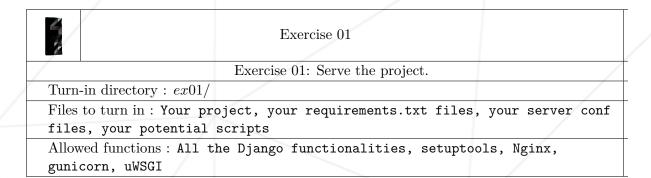
The ALLOWED\_HOSTS variable must NOT be set on ['\*'].

It would be a pity not get get points for the following exercises just because you've failed this one, right?

You must NOT turn-in the file created by your collect static.

#### Chapter V

#### Exercise 01



Your project is done? Well, serve it!

To do so, use a "real" serveur (Nginx), DON'T use Django's development server.

Start serving the project itself, without minding assets (fichiers MEDIA, STATIC...) for the moment.

You can create little scripts to ease your management of servers. However, all your processes will have to run in the background for this whole day.

# Chapter VI

## Exercise 02

	Exercise 02	
	Exercise 02: With "STATIC" files.	
Turn	n-in directory: $ex02/$	
Files	to turn in: Your project, your requirements.txt files, your server con	f
file	es, your potential scripts	
Allov	wed functions: All the Django functionalities, setuptools, Nginx,	
guni	corn, uWSGI	

No suspens here: make sure you serve your STATIC elements. Your potential style sheets for instance.

### Chapter VII

#### Exercise 03

	Exercise 03	
	Exercise 03: With "MEDIA" files.	
Turn	-in directory : $ex03/$	
Files file	${ m to} \ { m turn} \ { m in} : { m Your} \ { m project}$ , ${ m your} \ { m requirements.txt} \ { m files}$ , ${ m your} \ { m server} \ { m co}$	nf
	wed functions: All the Django functionalities, setuptools, Nginx, corn, uWSGI	

The same, with your MEDIA files this time around.

At this stage, you should have a conf Nginx allowing you to serve the whole project with the variable DEBUG set to False, with STATIC and MEDIA management. You should especially make sure all the details are working.

## Chapter VIII

#### Exercise 04

	Exercise 04	
	Exercise 04: In HTTPS.	/
Turn-in dire	ctory: ex04/	
Files to turn	in: Your project, your requirements.txt files, you	ur server conf
	ctions: All the Django functionalities, setuptools, uWSGI, SSL	Nginx,

For this last exercise, have all the traffic transit towards your HTTPS website. You will open a port fo the HTTP and another for the HTTPS. If you aim at the HTTP port, you will be automatically redirected towards the HTTPS port.

A warning message from the browser should not be disturbing in the context of this exercise if you use a self-signed certificate. (As long as you can add an exception).