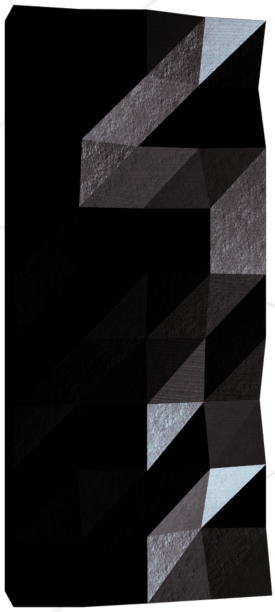

D08 - Setting production in motion.

Last step before completion!

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

Chapter I

Preamble

This will be a short preamble.

...

Done.

Chapter II

Ocaml piscine, general rules

- Every output goes to the standard output, and will be ended by a newline, unless specified otherwise.
- The imposed filenames must be followed to the letter, as well as class names, function names and method names, etc.
- Unless otherwise explicitly stated, the keywords `open`, `for` and `while` are forbidden. Their use will be flagged as cheating, no questions asked.
- Turn-in directories are `ex00/`, `ex01/`, ..., `exn/`.
- You must read the examples thoroughly. They can contain requirements that are not obvious in the exercise's description.
- Since you are allowed to use the `OCaml` syntaxes you learned about since the beginning of the piscine, you are not allowed to use any additional syntaxes, modules and libraries unless explicitly stated otherwise.
- The exercices must be done in order. The graduation will stop at the first failed exercice. Yes, the old school way.
- Read each exercise FULLY before starting it! Really, do it.
- The compiler to use is `ocamlopt`. When you are required to turn in a function, you must also include anything necessary to compile a full executable. That executable should display some tests that prove that you've done the exercise correctly.
- Remember that the special token `";;"` is only used to end an expression in the interpreter. Thus, it must never appear in any file you turn in. Regardless, the interpreter is a powerfull ally, learn to use it at its best as soon as possible!
- The subject can be modified up to 4 hours before the final turn-in time.
- In case you're wondering, no coding style is enforced during the `OCaml` piscine. You can use any style you like, no restrictions. But remember that a code your peer-evaluator can't read is a code he or she can't grade. As usual, big functions are a weak style.
- You will NOT be graded by a program, unless explictly stated in the subject. Therefore, you are given a certain amount of freedom in how you choose to do the

exercises. However, some piscine day might explicitly cancel this rule, and you will have to respect directions and outputs perfectly.

- Only the requested files must be turned in and thus present on the repository during the peer-evaluation.
- Even if the subject of an exercise is short, it's worth spending some time on it to be absolutely sure you understand what's expected of you, and that you did it in the best possible way.
- By Odin, by Thor! Use your brain!!!

Chapter III

Today's specific rules

- 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 : *ex*00/	
Files to turn in : `Your project, the sources for your application, your requirements.txt file`	
Allowed 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 collectstatic.

Chapter V

Exercise 01

	Exercise 01
Exercise 01: Serve the project.	
Turn-in directory : $ex01/$	
Files to turn in : `Your project, your requirements.txt files, your server conf files, your potential scripts`	
Allowed functions : `All the Django functionalities, setuptools, Nginx, gunicorn, uWSGI`	

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-in directory : $ex02/$	
Files to turn in : `Your project, your requirements.txt files, your server conf files, your potential scripts`	
Allowed functions : `All the Django functionalities, setuptools, Nginx, gunicorn, 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 to turn in : `Your project, your requirements.txt files, your server conf files`	
Allowed functions : `All the Django functionalities, setuptools, Nginx, gunicorn, 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 directory : $ex04/$	
Files to turn in : `Your project, your requirements.txt files, your server conf files`	
Allowed functions : `All the Django functionalities, setuptools, Nginx, gunicorn, uWSGI, SSL`	

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).