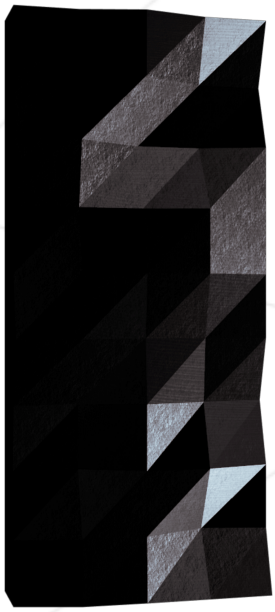

D04 - Python-Django Training

On with the serious business.

Summary: After Python, Django!

Contents

Chapter I

Preamble

Here is a list of monsters you may encounter wandering a dungeon in the land of Fangh:

- any kind of undead.
- giant spiders.
- orcs and goblins.
- trolls in the underground.
- sorcerers.
- cursed warriors.
- giant rats.
- an oil bottle.
- toilet paper.
- two sponges
- raviolis.

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

- All paths relative to an application must be defined in a `urls.py` file located in a folder of this application.
- Any form (derived class of `django.forms.Form`) must be located in the `forms.py` application's file it is related to.
- Each displayed page must be properly formatted (a doctype, couples of `html` tags, `body`,`head` must be included), proper management of special characters, no strange display.
- Today, you will use Django's default development server provided with the `manage.py` utility.
- Only the specifically requested URLs must return a page without any error. Hence, if just a `/ex00` is requested, `/ex00foo` must return a 404 error.
- Requested URLs must work with or without end slash. Hence, if `/ex00` is requested, `/ex00` and `/ex00/` both must work.

Chapter IV

Exercise 00

	Exercise 00
Exercise 00: First static page.	
Turn-in directory : $ex00/$	
Files to turn in : `requirements.txt, files and folders of your project and application`	
Allowed functions : `All Django functionalities.`	

With this exercise, you will make your first static page with Django.

Create a `virtualenv` with `python3`, install `Django`, create a `requirements.txt` file containing the project's dependancies (what's installed in the virtualenv with `pip`) at the root of the repo.

Start a project `d04`.

Start an application `ex00`.

Create a page that will searchable at the following URL `/ex00` of your website. In this page, gather all the informations about the whole `Markdown` syntax and give this page the title "`Ex00: Markdown Cheatsheet.`".

The used template will be named `index.html`.

Chapter V

Exercise 01

	Exercise 01
Exercise 01: A few more pages.	
Turn-in directory : *ex*01/	
Files to turn in : `requirements.txt, files and folders of your project and application.`	
Allowed functions : `All Django functionalities.`	

Create the following pages in a second application `ex01`:

- **Title :** "Ex01: Django, framework web."

  URL: `/ex01/django`.

  Description: In this page, briefly introduce Django and its history.

- **Title:** "Ex01: Display process of a static page."

  URL: `/ex01/display`

  Description: In this page, describe the process causing the display of a static web page from a simple template going through a view, from the request to the answer.

- **Title:** "Ex01: Template engine."

  URL: `/ex01/templates`

  Description: In this page, describe the functioning of Django's default template engine as well as the functioning of:

  - Blocks.
  - Loops`for ... in`.
  - `if` control structures.
  - The display of the passing context variables.

The principle of the *don't repeat yourself* is part of the Django's philosophy. To respect this principle, create the required pages using a base template named `base.html`.

The `base.html` template must include:

- A `content` block in the `body`.
- A `style` block in the `head`.
- A `title` block in the `head`.

Also create a `nav.html` template containing a navigation bar listing the links to each of the exercise's pages.

All the pages of this exercise must be based on the `base.html` template. The `nav.html` template must be included in each of your pages.

Also create 2 files: `style1.css` and `style2.css`. The first one will define the text color as *blue*, the other one will be *rouge*. You will have to use `style1.css` in every page, except in the "Template engine" page, where you will use the `style2.css`.

`You must use each style sheet just once in all your templates for this exercise.`

`You will have to use collectstatic during evaluation.`

Chapter VI

Exercise 02

	Exercise 02
Exercise 02: First form.	
Turn-in directory : $ex02/$	
Files to turn in : `requirements.txt, files and folders of your project and application.`	
Allowed functions : `All Django functionalities.`	

Create a page accessible at the URL `/ex02` in a new `ex02` application.

This page will contain 2 parts:

- A form with a text field and a `Submit` button.
- A part that contains input history.

Here is the rule regarding the form:

- You must `NOT` hard code the form's field. You will use the `django.forms.Form` class provided by Django to create a form. You will pass it as `context` to the template's render function.

Here are the rules for the history:

- The input history will start empty.
- For each text submission in the form, you will have to:
    - List the input and its timestamp at the end of a `logs` file. If the file doesn't exist, it must be automatically created. This file must be created in the folder or the `ex02` application.
    - Add the input to the page's history with the submission's time and date.
- The path to the `logs` file (file name included) must be defined in the project's `settings.py` file. Hence, each submitted entry will be included in the page's history and in the `log` file, preceded by its timestamp.
- The data must also be persistent. If the development server must restart somehow, the data must not be lost.
  As a rule, if you redisplay the page, the data are redisplayed as long as they're saved.

Chapter VII

Exercise 03

	Exercise 03
Exercise 03: Fifty shades of bic.	
Turn-in directory : *ex*03/	
Files to turn in : `requirements.txt, files and folders of your project and application.`	
Allowed functions : `All Django functionalities.`	

Create a final application `ex03`.

Display a page containing a 4 columns x 51 lines table (one line will be dedicated to the column's name).

You will access this page at this URL `/ex03`.

Each table column will have a different color: *noir*, *rouge*, *bleu* and *vert*.

The table boxes must have the following specifications:

Height: 40 pixels.
Width: 80 pixels.
Background color: a color shade that matches the column.

The table must display the shade of each color as to obtain a 50 lines shading.

You must observe the following instructions:

- Your template must `NOT` include the colors in hard. The different shades must be generated in a view and must all be different.
- A total of 4 tag couples `<td></td>` are authorized in your templates.
- A total of 4 tag couples `<th></th>` are authorized in your templates.
- A table must be formatted as follows:
  - A tag couple `<th></th>` per box for the column names.
  - A tag couple `<tr></tr>` per color shade line.
  - A tag couple `<td></td>` per color shade box.