Training Python-Django - 3

Final

Summary: *Today, we're gonna learn how to use AJAX and Websockets with* `Django`.

Version: 1.1

Contents

Chapter I

Preamble

`Chat`

The cat (*Felis catus*) is a small carnivorous mammal. It is the only domesticated species in the family Felidae and often referred to as the domestic cat to distinguish it from wild members of the family. The cat is either a house cat, a farm cat or a feral cat; latter ranges freely and avoids human contact. Domestic cats are valued by humans for companionship and for their ability to hunt rodents. About 60 cat breeds are recognized by various cat registries.
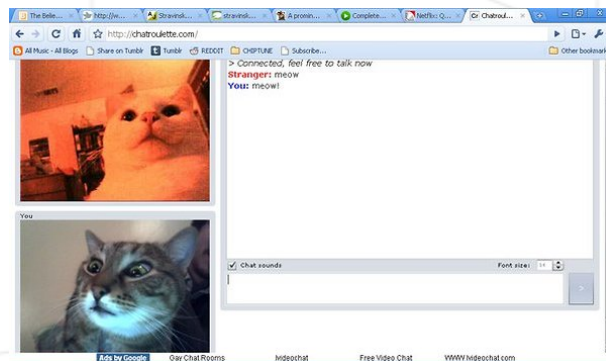
Source.

Figure I.1: Cat chatting on chatroulette.

Why do we even mention cats here you'll ask? Well, all these subjects were written by French writers and in french, a cat is called a "Chat". There's a pun, here, see? Maudits Français!

Chapter II

General rules

- Your project must be realized in a virtual machine.

- Your virtual machine must have all the necessary software to complete your project. These softwares must be configured and installed.

- You can choose the operating system to use for your virtual machine.

- You must be able to use your virtual machine from a cluster computer.

- You must use a shared folder between your virtual machine and your host machine.

- During your evaluations you will use this folder to share with your repository.

- Your functions should not quit unexpectedly (segmentation fault, bus error, double free, etc) apart from undefined behaviors. If this happens, your project will be considered non functional and will receive a `0` during the evaluation.

- We encourage you to create test programs for your project even though this work **won't have to be submitted and won't be graded**. It will give you a chance to easily test your work and your peers' work. You will find those tests especially useful during your defence. Indeed, during defence, you are free to use your tests and/or the tests of the peer you are evaluating.

- Submit your work to your assigned git repository. Only the work in the git repository will be graded. If Deepthought is assigned to grade your work, it will be done after your peer-evaluations. If an error happens in any section of your work during Deepthought's grading, the evaluation will stop.

Chapter III

Today's specific rules

- The only Javascript library you can use is `JQuery`.

- Your turn-in will be have the form of a unique `Django` project. It won't be divided in exercises, as usual. Each of them add a fucntionality to the project. This functionality and its implementation will be graded this time.

- You must leave the default adminstration application.

- Along with your project, you must turn-in a requirement.txt file (with the 'pip freeze' command) listing the libraries your project will need to be running.

Chapter IV

Exercise 00

	Exercise 00
	Exercice 00: AjAX my formulah!
Turn-in directory : *ex00/*	
Files to turn in :	
Allowed functions : `None`	

Create a new project named `d09`. In this project, create an application named `account`.

The goal of this exercise is to design a connection/disconnection system communicating only thanks to `AJAX`.

In this application , you will implement the `127.0.0.1:8000/account` URL that must lead to a page that can have two diffrent behaviors depending on the context:

- The user is not connected: the page must display a standard connection form (login, password). The thing is the communication with the server must only use AJAX and must be a `POST` type to get connected.

  If the form is not valid, the error(s) must appear on the page.

  Of the form is valid, it must disapear and adopt a new behavior.

  This, of course, without the page being refreshed.

- The user is connected already: the page must display the following text: `"Logged as <user>"`, <user> being replaced by the name used by the user to get connected as well as `Logout` button allowing disconnection.

  This button must communicate with the server via AJAX and the 'POST' method.

Once logged out, the text and button must not show on the page anymore and the latter must adopt the other behavior.

The page must never have been refreshed.

If the page is 'manually' refreshed, it must return to the behavior it had before refreshing (that does not include the error displaying).

You can use bootstrap.

`AuthenticationForm, for free!`

Chapter V

Exercise 01

![]	Exercise 01

Exercice 01: Basic chat
Turn-in directory : *ex01/*
Files to turn in :
Allowed functions : `None`

Create an application named `'chat'`.

In this application, you must create a page displaying 3 links. Each of them must lead to a different `'chatrooms'`.

The names of these rooms must be in database. You must create a suitable model.

Each of these links must lead to another page containing a standard functional chat. Each chat must have the following specification:

- It must use `'jquery'` as sole frontend library as well as the `Websockets` to communicate with the server.(no AJAX)

- It's only available to connected users.

- The name of the chat must appear somewhere.

- Several users must be able to connect (*just in case...*).

- A user can post a message (*you had guessed, right?*).

- A message sent by a user must be visible by all the users who have joined the chatroom (*everyone knows what a chatroom is, right? Haven't you read that preamble?*).

- Messages must appear in the bottom and be displayed in ascending order (*that one's for you, by the heater, right.*), along with the name of the user that posted them.

- Messages must not disappear. A message must not replace a previous one. The messages order must not change.

- When a user joins the chatroom, the message '`<username> has joined the chat`' must appear for all users to see, including the one who just joined. <username> is replaced by said user's name of course.

Chapter VI

Exercise 02

	Exercise 02
	Exercice 02: History
Turn-in directory : *ex02/*	
Files to turn in :	
Allowed functions : `None`	

In this exercise, you will improve your chat adding a message history to it.

When a new user joins the chatroom, they must see the last three messages that have been posted `on this chatroom`, top down, oldest to newest.

Once again, you can only use `JQuery` as frontend libraries and `Websockets` to communicate with the server.

Chapter VII

Exercise 03

	Exercise 03
	Exercice 03: Userlist
Turn-in directory : *ex03/*	
Files to turn in :	
Allowed functions : `None`	

In this exercise, you will improve your chat again adding a connected userlist, this time. AND it will update by itself.

When the user joins the chatroom, he must be able to access the list of connected users (and he must appear in the list).

This userlist must be clearly set aside the messages list (other <div> or other `html` container).

When a user joins a chatroom, their name must appear in the list, along with other connected users.

When a user leaves the chatroom, their name must disappear from the list of connected users and the message '`<username> has left the chat`' must appear after the posted messages. (<username> will be the name of the user who just left)

Once again, you can only use `JQuery` as frontend libraries and `Websockets` to communicate with the server.

> 💡 Before anything, try to build a functional logic. We're not looking for optimization... yet.

Chapter VIII

Exercise 04

	Exercise 04

Exercise 04: Scroll
Turn-in directory : *ex04/*
Files to turn in :
Allowed functions : `None`

Make your chat presentable setting your message list in a fixed size container. If the number of messages exceed the container, they must disappear on top and a scroll bar must appear on the side.

Besides,the scroll bar must always appear with the cursor down so the last messages always show first.

Chapter IX

Submission and peer-evaluation

Turn in your assignment in your `Git` repository as usual. Only the work inside your repository will be evaluated during the defense. Don't hesitate to double check the names of your folders and files to ensure they are correct.

> The evaluation process will happen on the computer of the evaluated group.