



# Formation Python-Django - 3

Final

*Résumé: Aujourd'hui nous allons découvrir comment se servir d'AJAX et des Websockets avec Django.*

*Version: 1*

# Table des matières

<b>I</b>	<b>Préambule</b>	<b>2</b>
<b>II</b>	<b>Règles communes</b>	<b>3</b>
<b>III</b>	<b>Règles spécifiques de la journée</b>	<b>4</b>
<b>IV</b>	<b>Exercice 00</b>	<b>5</b>
<b>V</b>	<b>Exercice 01</b>	<b>7</b>
<b>VI</b>	<b>Exercice 02</b>	<b>9</b>
<b>VII</b>	<b>Exercice 03</b>	<b>10</b>
<b>VIII</b>	<b>Exercice 04</b>	<b>11</b>
<b>IX</b>	<b>Rendu et peer-évaluation</b>	<b>12</b>

# Chapitre I

## Préambule

### Chat

Le chat domestique (*Felis silvestris catus*) est la sous-espèce issue de la domestication du chat sauvage, mammifère carnivore de la famille des félidés. Il est l'un des principaux animaux de compagnie et compte aujourd'hui une cinquantaine de races différentes reconnues par les instances de certification. Dans de nombreux pays, le chat entre dans le cadre de la législation sur les carnivores domestiques à l'instar du chien et du furet.

[Source.](#)

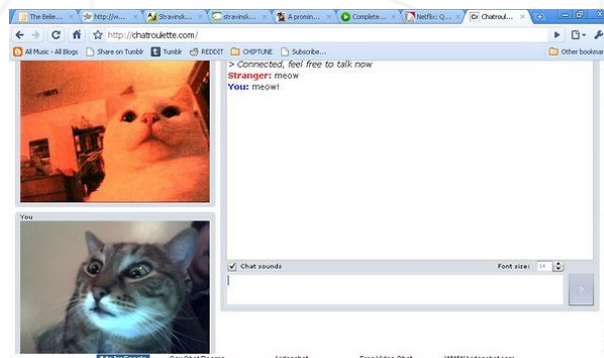


FIGURE I.1 – Chat chattant sur chatroulette.



Aucun exercice lors de cette journée ne parle de ce chat là . Je préfère préciser avant qu'un doute ne s'installe. N'allez pas croire que je doute de votre intelligence hein ? Je préfère juste anticiper ... pour ceux du fond ... près du radiateur ... c'est toujours ceux du fond de toute façon. \*soupire\*

# Chapitre II

## Règles communes

- Votre projet doit être réalisé dans une machine virtuelle.
- Votre machine virtuelle doit avoir tout les logiciels necessaire pour réaliser votre projet. Ces logiciels doivent être configurés et installés.
- Vous êtes libre sur le choix du systmème d'exploitation à utiliser pour votre machine virtuelle.
- Vous devez pouvoir utiliser votre machine virtuelle depuis un ordinateur en cluster.
- Vous devez utiliser un dossier partagé entre votre machine virtuelle et votre machine hôte.
- Lors de vos évaluations vous allez utiliser ce dossier partager avec votre dépôt de rendu.
- Vos fonctions de doivent pas s'arrêter de manière inattendue (segmentation fault, bus error, double free, etc) mis à part dans le cas d'un comportement indéfini. Si cela arrive, votre projet sera considéré non fonctionnel et vous aurez 0 au projet.
- Nous vous recommandons de créer des programmes de test pour votre projet, bien que ce travail **ne sera pas rendu ni noté**. Cela vous donnera une chance de tester facilement votre travail ainsi que celui de vos pairs.
- Vous devez rendre votre travail sur le git qui vous est assigné. Seul le travail déposé sur git sera évalué. Si Deepthought doit corriger votre travail, cela sera fait à la fin des peer-evaluations. Si une erreur se produit pendant l'évaluation Deepthought, celle-ci s'arrête.


# Chapitre III

## Règles spécifiques de la journée

- La seule librairie javascript que vous avez la permission d'utiliser est JQuery
- Votre rendu prendra la forme d'un unique projet Django. Il n'aura pas le découpage habituel en exercices. Chacun d'entre eux rajoutent au projet une fonctionnalité. C'est celle-ci, ainsi que son implémentation qui seront notés.
- Vous devez laisser l'application d'administration par défaut.
- Vous devez rendre avec votre projet un fichier requirement.txt (via la commande 'pip freeze') listant les librairies nécessaires au fonctionnement de votre projet.

# Chapitre IV

## Exercice 00

	Exercice : 00
Exercice 00 : Ajax my formulah !	
Dossier de rendu : ex00/	
Fichiers à rendre :	
Fonctions Autorisées :	

Créez un nouveau projet nommé d09 et dans ce projet une application nommée **account**.

L'objectif de cet exercice est de concevoir un système de connection/ déconnection communiquant uniquement grâce à **AJAX**.

Vous devez implémenter dans cette application l'url **127.0.0.1:8000/account** qui doit renvoyer une page qui peut avoir deux comportements différents selon deux cas de figure :

- L'utilisateur n'est pas connecté : La page doit afficher un formulaire de connection standard (login, mot de passe), si ce n'est que la communication avec le serveur pour valider le formulaire doit utiliser **AJAX** uniquement et doit être de type **POST**.

Si le formulaire n'est pas valide, le ou les erreurs doivent être affichées sur la page.

Si le formulaire est valide, celui-ci doit disparaître et adopter l'autre comportement.

Tout ça sans que la page ne soit à aucun moment rafraîchie.

- L'utilisateur est déjà connecté : La page doit afficher la phrase "**Logged as <user>**", **<user>** étant à remplacer par le nom avec lequel l'utilisateur est connecté, ainsi qu'un bouton **Logout** permettant de se déconnecter.

Ce bouton doit communiquer avec le serveur via **AJAX** et avec la méthode '**POST**'.

Une fois déloggé, la phrase ainsi que le bouton doivent disparaître de la page et

celle ci doit adopter l'autre. comportement.

Tout ça sans que la page ne soit à aucun moment rafraîchie.

Dans le cas ou la page est rafraîchie 'manuellement', celle-ci doit retrouver le comportement dans lequel vous l'avez laissée (Cela n'inclut pas l'affichage des erreurs).


Vous pouvez utiliser bootstrap.



AuthenticationForm, c'est cadeau !

# Chapitre V

## Exercice 01

	Exercice : 01
Exercice 01 : Chat de base	
Dossier de rendu : <i>ex01/</i>	
Fichiers à rendre :	
Fonctions Autorisées :	

Créez une nouvelle application nommée '**chat**'.

Dans cette application vous devez créer une page affichant trois liens qui doivent chacun mener à une '**chatrooms**' différentes.

Le nom de ces rooms doit être en base de donnée, vous devez donc créer un modèle adéquat.

Chacun de ces liens doit mener à une autre page contenant un chat standard fonctionnel. Chaque chat doit posséder les caractéristiques suivantes :

- Il doit utiliser '**jquery**' comme unique librairie frontend ainsi que les **Websockets** pour communiquer avec le serveur. (pas d'**AJAX**)
- Il n'est accessible qu'aux utilisateurs connectés.
- Le nom du chat doit apparaître quelque part.
- Plusieurs utilisateurs doivent pouvoir s'y connecter (*dès fois que vous en doutiez ...*).
- Il est possible pour un utilisateur de poster un message ( *mais vous le saviez aussi non ?* ).
- Un message envoyé par un utilisateur doit être visible par tous les autres d'une même chatroom (*tout le monde sait ce qu'est un chat n'est ce pas ? Vous avez lu le préambule ?*).
- Les messages doivent s'afficher de haut en bas, du plus ancien au plus récent (*ça*




*c'est pour les originaux la bas au fond ... c'est toujours ceux du fond de toute façon.), accompagné du nom de l'utilisateur qui l'a posté.*

- Les messages ne doivent pas disparaître, un message ne doit pas en remplacer un autre, l'ordre des messages ne doit pas bouger.
- Lorsqu'un utilisateur se connecte, le message '`<username> has joined the chat`' doit apparaître pour tous les utilisateurs y compris lui-même. `<username>` est a remplacé par le nom de l'utilisateur en question.

# Chapitre VI

## Exercice 02

	Exercice : 02
Exercice 02 : Historique	
Dossier de rendu : <i>ex02/</i>	
Fichiers à rendre :	
Fonctions Autorisées :	


Dans cet exercice, vous allez améliorer votre chat en lui offrant un historique des messages.

Lorsqu'un nouvel utilisateur se joint à une chatroom, il doit voir s'afficher les trois derniers messages qui ont été postés **sur ce chatroom**, de haut en bas, du plus ancien au plus récent.

Encore une fois seuls **JQuery** comme librairie frontend et les **Websockets** pour communiquer avec le serveur sont permis.

# Chapitre VII

## Exercice 03

	Exercice : 03
Exercice 03 : Userlist	
Dossier de rendu : <i>ex03/</i>	
Fichiers à rendre :	
Fonctions Autorisées :	

Dans cet exercice vous allez encore améliorer votre chat en lui offrant une liste des utilisateurs connectés qui se met à jour toute seule comme une grande.

Lorsque l'utilisateur se connecte à une chatroom, il doit avoir accès de suite à la liste des utilisateurs connectés (dont lui-même).

Cette liste d'utilisateur doit être distincte visuellement de la liste des messages (autre `<div>` ou autre conteneur `html`).

Lorsqu'un utilisateur se connecte à une chatroom, son nom doit apparaître dans la liste des autres utilisateurs connectés.

Lorsqu'un utilisateur quitte une chatroom, son nom doit disparaître de la liste des autres utilisateurs connectés et le message '`<username> has left the chat`' doit apparaître (`<username>` à remplacer par le nom de l'utilisateur en question) à la suite des messages postés.


Encore une fois seuls **JQuery** comme librairie frontend et les **Websockets** pour communiquer avec le serveur sont permis.



Cherchez avant tout à construire une logique fonctionnelle.  
L'objectif n'est pas l'optimisation

# Chapitre VIII

## Exercice 04

	Exercice : 04
Exercice 04 : Scrool	
Dossier de rendu : <i>ex04/</i>	
Fichiers à rendre :	
Fonctions Autorisées :	

Rendez votre chat présentable en mettant la liste des messages dans un conteneur d'une hauteur et d'une largeur fixe. Si le nombre de messages dépasse cette hauteur, les messages en trop disparaissent et une barre de scroll permet de les faire défiler.

De plus, la barre de scroll doit toujours se trouver en bas, de manière à ce que les derniers messages soient toujours en vue.

# Chapitre IX

## Rendu et peer-évaluation

Rendez votre travail dans votre dépôt `Git` comme d'habitude. Seul le travail présent dans votre dépôt sera évalué en soutenance. Vérifiez bien les noms de vos dossiers et de vos fichiers afin que ces derniers soient conformes aux demandes du sujet.



L'évaluation se déroulera sur l'ordinateur du groupe évalué.