



D09 - Formation Python-Django

Django - Ajax - Websockets

Damien Cojan dcojan@student.42.fr
42 Staff pedago@staff.42.fr

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

Table des matières

| | | |
|-------------|---|-----------|
| I | Préambule | 2 |
| II | Consignes | 3 |
| III | Règles spécifiques de la journée | 5 |
| IV | Exercice 00 | 6 |
| V | Exercice 01 | 8 |
| VI | Exercice 02 | 10 |
| VII | Exercice 03 | 11 |
| VIII | Exercice 04 | 12 |

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.](#)

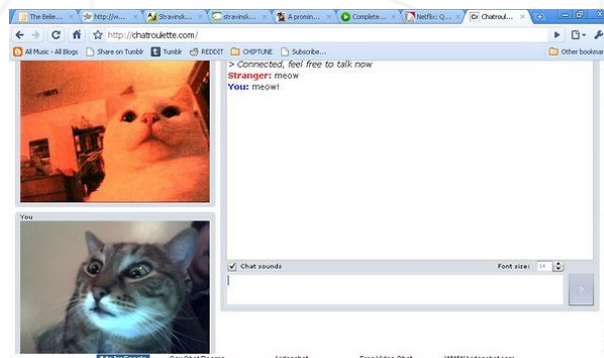


FIGURE I.1 – Chat chattant sur chatroulette.



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

Consignes

- Seule cette page servira de référence : ne vous fiez pas aux bruits de couloir.
- Le sujet peut changer jusqu'à une heure avant le rendu.
- Si aucune information contraire n'est explicitement présente, vous devez partir du principe que les versions des langages et framework utilisés sont les suivantes (ou ultérieures) :
 - Python 3
 - HTML5
 - CSS 3
 - Javascript ES6
 - Django 1.9
 - psycopg2 2.6
- Sauf indication contraire dans le sujet, les fichiers en python de chaque exercice sur Python seul (d01, d02 et d03) doivent comporter à leur fin un bloc `if __name__ == '__main__':` afin d'y insérer le point d'entrée dans le cas d'un programme, ou des tests dans le cas d'un module.
- Sauf indication contraire dans le sujet, chaque exercice des journées portant sur Django aura sa propre application dans le projet à rendre pour des raisons pédagogiques.
- Les exercices sont très précisément ordonnés du plus simple au plus complexe. En aucun cas nous ne porterons attention ni ne prendrons en compte un exercice complexe si un exercice plus simple n'est pas parfaitement réussi.
- Attention aux droits de vos fichiers et de vos répertoires.
- Vous devez suivre la procédure de rendu pour tous vos exercices : seul le travail présent sur votre dépôt GIT sera évalué en soutenance.
- Vos exercices seront évalués par vos camarades de piscine.
- Vous ne devez laisser dans votre répertoire aucun autre fichier que ceux explicitement spécifiés par les énoncés des exercices.
- Sauf indication contraire dans le sujet vous ne devez pas inclure dans votre rendu :

- Les dossiers `__pycache__`.
- Les éventuelles migrations.
Attention, il vous est tout de même conseillé de rendre le fichier `migrations/__init__.py`, il n'est pas nécessaire mais simplifie la construction des migrations.
Ne pas ajouter ce fichier n'invalidera pas votre rendu mais vous *devez* être capables de gérer vos migrations en correction dans ce cas.
- Le dossier créé par la commande `collectstatic` de `manage.py` (avec pour chemin la valeur de la variable `STATIC_ROOT`).
- Les fichier en bytecode Python (Les fichiers avec une extension en `.pyc`).
- Les fichiers de base de donnée (notamment avec `sqlite`).
- Tout fichier ou dossier devant ou pouvant être créé par le comportement normal du travail rendu.
Il vous est recommandé de modifier votre `.gitignore` afin d'éviter les accidents.
- Lorsque vous devez obtenir une sortie précise dans vos programmes, il est bien entendu interdit d'afficher une sortie précalculée au lieu de réaliser l'exercice correctement.
- Vous avez une question ? Demandez à votre voisin de droite. Sinon, essayez avec votre voisin de gauche.
- Votre manuel de référence s'appelle `Google / man / Internet /`
- Pensez à discuter sur le forum Piscine de votre Intra !
- Lisez attentivement les exemples. Ils pourraient bien requérir des choses qui ne sont pas autrement précisées dans le sujet...
- Par pitié, par Thor et par Odin ! Réfléchissez nom d'une pipe !


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 `requirements.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 : <i>ex00/</i> | |
| Fichiers à rendre : Les fichiers nécessaires pour votre application | |
| Fonctions Autorisées : Voir consigne | |
| Remarques : n/a | |

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 rafraîchie... à aucun moment.

- 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 délogué, 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 rafraîchie... toujours à aucun moment

Dans le cas où 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 : Les fichiers nécessaires pour votre application | |
| Fonctions Autorisées : Voir consigne | |
| Remarques : n/a | |

Créez une nouvelle application nommée 'chat'.

Dans cette application vous devez créer une page affichant trois liens qui doivent chacun mener à une 'chatroom' différente.

Le nom de ces rooms doit être en base de données, vous devrez 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 à remplacer 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 : Les fichiers nécessaires pour votre application | |
| Fonctions Autorisées : Voir consigne | |
| Remarques : n/a | |


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 cette 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 : Les fichiers nécessaires pour votre application | |
| Fonctions Autorisées : Voir consigne | |
| Remarques : n/a | |

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 : Scroll | |
| Dossier de rendu : <i>ex04/</i> | |
| Fichiers à rendre : Les fichiers nécessaires pour votre application | |
| Fonctions Autorisées : Voir consigne | |
| Remarques : n/a | |

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.