4. Votre premier projet Django !

Une partie de ce chapitre s'inspire du tutoriel des Geek Girls Carrots (https://github.com/ggcarrots/django-carrots).

Des morceaux de ce chapitre sont inspirés du tutoriel django-marcador, disponible sous licence Creative Commons Attribution-ShareAlike 4.0 International. Le tutoriel django-marcador a été créé par Markus Zapke-Gründemann et al.

Nous allons créer un petit blog !

La première étape consiste à démarrer un nouveau projet Django. En gros, cela veut dire que nous allons lancer quelques scripts fournis par Django qui vont créer un squelette de projet Django. Il s'agit de fichiers et de dossiers que nous utiliserons par la suite.

Il y existe certains fichiers et dossiers dont les noms sont extrêmement importants pour Django. Il ne faut pas renommer les fichiers que nous sommes sur le point de créer. Ce n'est pas non plus une bonne idée de les déplacer. Django a besoin de maintenir une certaine structure pour retrouver les éléments importants.

Create project: OS X or Linux

- Retournons à la création de notre premier projet. Tapez la commande suivant dans votre console MacOS ou Linux. N'oubliez pas le point '.' à la fin !

```
~/djangology$ django-admin startproject mysite .
```

- Le point '.' est très important : c'est lui qui permet de dire au script d'installer Django dans votre répertoire courant (le point '.' est une référence abrégée à celui-ci).

Note : lorsque vous tapez la commande précédente dans votre console, vous ne devez recopier que la partie qui commence par 'django-admin'. La partie '~/djangology$' montrée ici n'est qu'un exemple pour vous rappeler de taper la commande dans votre console.

Create project: Windows

- Sur Windows, vous devez taper la commander suivante. (N'oubliez pas le point '.' à la fin) :

```
C:\Users\Name\djangology> django-admin.exe startproject mysite .
```

- Le point '.' est très important : c'est lui qui permet de dire au script d'installer Django dans votre répertoire courant (le point '.' est une référence abrégée à celui-ci).

Note : lorsque vous tapez la commande précédente dans votre console, vous ne devez recopier que la partie qui commence par 'django-admin.exe'. La partie 'C:\Users\Name\djangology>' montrée ici n'est qu'un exemple pour vous rappeler de taper la commande dans votre console.

django-admin.py

django-admin.py est un script qui crée les dossiers et fichiers nécessaires pour vous. Vous devriez maintenant avoir une structure de dossier qui ressemble à celle-ci:

- djangology
  - manage.py
  - mysite
    - init.py
    - settings.py
    - urls.py
    - wsgi.py
  - requirements.txt

manage.py est un script qui aide à gérer ou maintenir le site. Entre autres, il permet notamment de lancer un serveur web sur notre ordinateur sans rien installer d'autre.

Le fichier settings.py contient la configuration de votre site web.

Vous vous souvenez de l'histoire du postier qui livre des lettres ? urls.py contient une liste de patterns d'urls utilisés par urlresolver.

Ignorons les autres fichiers pour l'instant, nous n'allons pas avoir besoin d'y toucher. La seule chose à retenir est qu'il ne faut pas les supprimer par accident !

Changer la configuration

Apportons quelques changements à mysite/settings.py. Ouvrez votre éditeur (vscode ou pycharm) et ouvrez le projet djangology (ou tout autre nom de dossier que vous avez défini lors de l'étape précédente). Note : Gardez à l'esprit que settings.py est un fichier ordinaire, comme les autres. Vous pouvez l'ouvrir depuis l'éditeur.

Ça serait sympa d'avoir l'heure correcte sur notre site Web. Allez sur la liste Wikipedia des fuseaux horaires et copiez votre fuseau horaire (TZ) (par exemple Europe/Berlin).

Dans settings.py, recherchez la ligne qui contient TIME_ZONE et modifiez-la pour choisir votre propre fuseau horaire. Par exemple :

mysite/settings.py

```
  TIME_ZONE = 'Europe/Berlin'
```

Un code de langue se compose de la langue, par exemple en pour l'anglais ou de pour l'allemand, et du code du pays, par exemple de pour l'Allemagne ou ch pour la Suisse. Si l'anglais n'est pas votre langue maternelle, vous pouvez ajouter votre code de langue afin que les boutons par défaut et les notifications de Django soient traduits. Vous auriez alors le bouton "Cancel" traduit dans la langue

que vous avez définie. Django est livré avec un grand nombre de traductions disponibles.

Si vous voulez changer la langue, modifiez le code de langue comme montré dans la ligne suivante :

mysite/settings.py

```
LANGUAGE_CODE = 'fr'
```

Nous allons avoir besoin aussi d'un chemin d'accès pour les fichiers statiques. (Nous allons découvrir les fichiers statiques et CSS plus tard dans le tutoriel) Allez à la fin du fichier et juste en dessous de STATIC_URL, ajoutez une nouvelle entrée STATIC_ROOT :

mysite/settings.py

```
STATIC_URL = '/static/'
STATIC_ROOT = os.path.join(BASE_DIR, 'static')
```

Lorsque DEBUG a valeur True et ALLOWED_HOSTS est vide, les noms d'hôte acceptés sont ['localhost', '127.0.0.1', '[::1]']. Notre nom d'hôte sur PythonAnywhere ne sera donc pas accepté une fois que notre application sera déployée. Pour éviter cela, nous allons changer le paramètre suivant :

mysite/settings.py

```
ALLOWED_HOSTS = ['127.0.0.1', '.pythonanywhere.com']
```

Note : Si vous utilisez un Chromebook, ajoutez cette ligne à la fin de votre fichier settings.py : MESSAGE_STORAGE = 'django.contrib.messages.storage.session.SessionStorage' Ajoutez .amazonaws.com à ALLOWED_HOSTS si vous utilisez cloud9

Configuration de la base de données

Il existe tout un tas de systèmes de gestion de bases de données qu'il est possible d'utiliser pour stocker les données de votre site. Nous allons va utiliser celui par défaut : sqlite3.

Il est déjà configuré dans cette partie de votre fichier mysite/settings.py:

mysite/settings.py

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}
```

Pour créer la base de donnée de notre blog, il faut lancer la commande suivante dans la console : ~/djangology$ python manage.py migrate (vous avez besoin d'être dans le dossier djangology qui contient le fichier manage.py). Si tout se passe bien, vous devriez voir quelque chose comme ça:

command-line

```
~/djangology$ python manage.py migrate
Operations to perform:
Apply all migrations: auth, admin, contenttypes, sessions
Running migrations:
Rendering model states... DONE
Applying contenttypes.0001_initial... OK
Applying auth.0001_initial... OK
Applying admin.0001_initial... OK
Applying admin.0002_logentry_remove_auto_add... OK
Applying contenttypes.0002_remove_content_type_name... OK
Applying auth.0002_alter_permission_name_max_length... OK
Applying auth.0003_alter_user_email_max_length... OK
Applying auth.0004_alter_user_username_opts... OK
Applying auth.0005_alter_user_last_login_null... OK
Applying auth.0006_require_contenttypes_0002... OK
Applying auth.0007_alter_validators_add_error_messages... OK
Applying auth.0008_alter_user_username_max_length... OK
Applying auth.0009_alter_user_last_name_max_length... OK
Applying sessions.0001_initial... OK
```

Et voilà ! Il ne reste plus qu'à lancer le serveur et voir si notre site web fonctionne !

Lancer le serveur web

Pour cela, vous avez besoin d'être dans le dossier qui contient le fichier manage.py (le dossier djangology). Dans votre console, vous pouvez lancer le serveur en tapant python manage.py runserver:

command-line

~/djangology$ python manage.py runserver

Si vous avez un Chromebook, utilisez plutôt la commande suivante :

Cloud 9

~/djangology$ python manage.py runserver 0.0.0.0:8080

Si vous utilisez Windows et que vous obtenez l'erreur UnicodeDecodeError, tapez plutôt cette commande :

command-line

~/djangology$ python manage.py runserver 0:8000

Ensuite, vous allez vérifier que votre site fonctionne. Pour cela, ouvrez votre navigateur (Firefox,

Chrome, Safari, Internet Explorer, ou n'importe quel autre), et entrez l'adresse suivante :

navigateur

http://127.0.0.1:8000/

Si vous utilisez un Chromebook et Cloud9, cliquez plutôt sur l'URL dans la fenêtre pop-up qui devrait figurer dans le coin supérieur droit de la fenêtre de commande où le serveur web est en cours d'exécution. Le URL ressemble à ceci :

navigateur

https:*<a bunch of letters and numbers>.vfs.cloud9.us-west-2.amazonaws.com Bravo ! Vous venez de créer votre premier site web, et de le lancer avec un serveur web ! C'est génial, non? Notez qu'une fenêtre de commande ne peut exécuter qu'une chose à la fois, et la fenêtre de commande que vous avez ouvert précédemment est en train d'exécuter le serveur web. Tant que le serveur web est en cours d'exécution et en attente des demandes entrantes, le terminal acceptera du nouveau texte mais ne l'exécutera pas. Pour taper et exécuter des nouvelles commandes pendant que le serveur web est en fonction, ouvrez une nouvelle fenêtre dans votre terminal et activez votre virtualenv. Pour reviser comment ouvrir une nouvelle fenêtre, rendez-vous au chapitre Introduction à la ligne de commande. Pour arrêter votre serveur web, retournez dans la fenêtre où il tourne et appuyez sur CTRL+C : gardez les boutons Control et C appuyés en même temps. (Sous Windows, vous devrez peut-être appuyer sur CTRL+Arrêt défil.) Prêt-e pour la suite ? Il est temps de créer du contenu!*