Chapitre 2
Création de site

1. Objectifs

Dans le chapitre précédent, les outils et l'infrastructure logicielle nécessaires à la création de site ont été mis en place. Ceci va permettre de poser les bases du projet.

Le choix du répertoire racine d'un site est libre, il n'y a aucune obligation en rapport avec Python ou Django. De même il n'y a pas à se placer sous la racine d'un serveur frontal (Apache, Nginx, etc.) et il vaut d'ailleurs mieux l'éviter pour ne pas s'exposer à un éventuel risque de visibilité du code à cause d'une faille de sécurité. Le présent exposé va être poursuivi avec une destination courte et expressive, choisie arbitrairement à `D:\dj`.

2. Création d'un projet

La mise en œuvre d'une infrastructure logicielle suppose le respect d'un cadre conventionnel de disposition des éléments, ou, dit en d'autres termes, mettre les choses là où l'infrastructure a prévu de les trouver. Plutôt que de tout devoir écrire soi-même, des assistants sont mis à disposition pour poser ce cadre initial, sur la base d'un gabarit, avec des valeurs par défaut supposées convenir dans la majorité des cas.

▶Créez un nouveau répertoire pour héberger le projet :

```
D:\>md dj
```

▶Créez un nouveau projet, selon l'une ou l'autre façon :

```
D:\>\Python37\Scripts\django-admin startproject mysite dj
```

Ou :

```
D:\>cd dj
D:\dj>\Python37\Scripts\django-admin startproject mysite .
```

Dans ce cas, notez bien la présence d'un caractère point isolé final.

Cette structure va être créée :

```
dj\
├─ manage.py
└─ mysite\
   ├─ __init__.py
   ├─ settings.py
   ├─ urls.py
   └─ wsgi.py
```

La commande `startproject` a, d'une part, un paramètre obligatoire pour nommer le projet et, d'autre part, un paramètre optionnel pour citer un chemin de répertoire, devant exister préalablement, pour accueillir le projet.

Le nom du projet est un choix libre, employer le mot `mysite` est juste une convention usuelle. En effet, il n'est pas nécessaire d'utiliser le nom du projet puisque le répertoire racine donne déjà son nom (`dj` dans le cas présent). De plus, lorsqu'on passe d'un site à un autre, il est plus facile de comprendre que les paquets désignés sous le préfixe `mysite` concernent spécifiquement le site courant, en contraste évident avec des paquets de provenance externe. L'éventuelle recopie de paquets d'un projet vers un autre est aussi facilitée puisqu'on évite ainsi l'effort fastidieux des mises en accord du nom.

Le choix est fait de citer explicitement le paramètre optionnel, car en son absence un premier répertoire sera créé en reprenant le nom du projet (s'il existe déjà, la commande échouera).

La commande minimale, que l'on voit habituellement dans les documentations et tutoriels, de la forme :

```
django-admin startproject mysite
```

aurait produit une structure telle que :

```
dj\
└─ mysite\
  ├─ manage.py
  └─ mysite\
     └─ ...
```

Comme on a naturellement tendance à s'être déjà placé dans un répertoire vierge, nouvellement créé pour être dédié au projet, la présence d'un premier répertoire `mysite` paraît superflue et on se demande quel peut être son rôle et si son nom doit être préservé. En réalité il ne s'agit que d'un conteneur, dont le nom et la présence n'ont pas d'importance pour Django. Il est donc permis de supprimer cet étage ou de le renommer.

Les fichiers arrivés sont :

- `manage.py` : le point d'entrée pour les commandes en mode ligne destinées à ce projet. Les commandes documentées pour `django-admin` s'appliquent à cet utilitaire puisqu'il joue le même rôle, mais du fait qu'il aménage son environnement, il a l'avantage de cibler ce projet implicitement, sans qu'il soit nécessaire de le préciser.
- `__init__.py` : le traditionnel fichier pour qualifier ce répertoire au rang d'un paquet Python. Ce fichier est ici vide.
- `settings.py` : les paramètres de configuration du projet pour diriger l'infrastructure logicielle.
- `urls.py` : les déclarations de routes du projet.
- `wsgi.py` : le point d'entrée à l'usage des serveurs web frontaux qui implémentent l'interface WSGI, permettant ainsi un déploiement aisé du projet.

3. Premier lancement du site

Afin de se rassurer sur la bonne situation des actions menées jusqu'à présent, il est permis à ce stade de tenter un lancement du serveur de développement de Django.

▶Passez la commande de lancement du serveur intégré :

```
D:\dj>py manage.py runserver
Performing system checks...

System check identified no issues (0 silenced).

You have 15 unapplied migration(s). Your project may not work
properly until you apply the migrations for app(s): admin, auth,
contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
<Mois JJ, AAAA - hh:mm:ss>
Django version 2.1.5, using settings 'mysite.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

On constate qu'à l'occasion de ce lancement, il est préalablement procédé à un balayage du projet, à la recherche d'erreurs usuelles. Il s'agit des mêmes contrôles que ceux de la commande `check`.

Le message d'avertissement à propos de migrations en attente d'application est attendu dans le contexte actuel. Il suffit de l'ignorer et il n'est pas nécessaire de passer la commande mentionnée.

Comme il est dit dans la sortie de la console et dans la documentation de la commande `runserver`, il ne s'agit aucunement d'un serveur utilisable pour un environnement de production.

C'est un outil dédié aux développeurs pour faciliter leur travail de conception, car il présente les avantages suivants :

– Il est disponible sur la machine locale, immédiatement et sans effort. L'interface réseau et le port sont par défaut à 127.0.0.1 et 8000, mais il est possible de spécifier d'autres valeurs en arguments. Par exemple : `manage.py runserver 192.168.0.10:8899` permet de servir aussi des requêtes provenant d'autres postes de travail que sa propre machine (192.168.0.10 étant dans cet exemple une adresse IP privée sur un segment de réseau local). On peut aussi employer l'adresse `0.0.0.0` pour écouter toutes les interfaces réseau de la machine. Son raccourci `0` mentionné par la documentation Django ne semble pas être supporté sur la plateforme Windows et le lancement échoue avec l'erreur « `Error: [Errno 11001] getaddrinfo failed` ». Une autre variante est d'employer le nom de la machine, par exemple `PC-Patrick:8001`, mais cela suppose un complément de configuration concernant `ALLOWED_HOSTS`, en rapport avec des protections de sécurité.
– Un changement dans les fichiers de code amorce automatiquement un rechargement du serveur, sans qu'il soit nécessaire de penser à l'arrêter et le relancer. Cette manipulation d'arrêt-relance reste quand même un passage obligé dans les cas d'ajouts de fichier. Il est aussi à noter que la surveillance a un coût en ressources système, car elle se réalise par la lecture toutes les secondes de la date de modification des fichiers. On peut ainsi constater à l'état de repos du serveur une consommation continuelle de CPU (processeur) de l'ordre de 4 à 7 %. L'argument optionnel `--noreload` permet de désactiver ce comportement et ainsi laisse l'occupation CPU à 0 % au repos, ce qui peut avoir un intérêt, par exemple pour un simple usage de démonstration ou pour épargner la consommation énergétique sur un portable.

Si nécessaire, il est possible d'avoir plusieurs serveurs actifs en même temps, a priori sur des projets séparés, à la seule condition de leur affecter des numéros de port différents.

Si on observe les répertoires du projet, on constate la présence d'un fichier supplémentaire à la suite du lancement :

```
dj\
├─ db.sqlite3
└─ ...
```

Rien n'a encore été établi relativement à la base de données, mais puisque l'infrastructure logicielle a la nécessité d'avoir une liaison avec une base, la configuration fournie par le gabarit prévoit le plus simple, c'est-à-dire une base `sqlite3` puisque sa forme se réduit à un simple fichier. La configuration a établi que le fichier porte ce nom et soit localisé à cet emplacement. Il est normal que le fichier soit encore totalement vide, aucune action de création de tables n'a eu lieu jusqu'à présent.

▶Allez sur le site http://localhost:8000 pour y constater la page servie :