Django : un framework pour le Web

Delphine Salvy

Séminaire Cr@ns

20 novembre 2018

I. Le fonctionnement de Django

Le modèle MTV

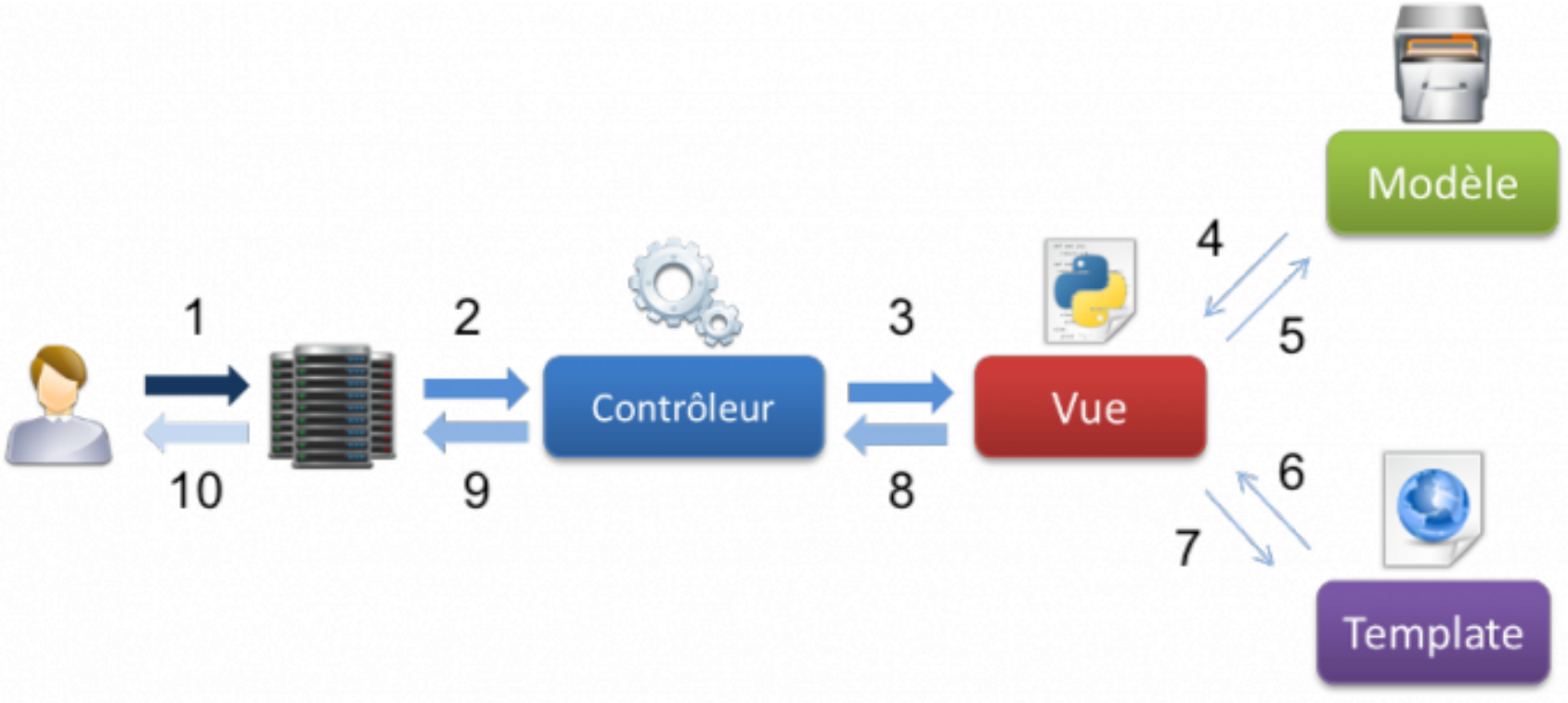
Django se base sur le modèle MTV (Model - Template - View), une variante du modèle MVC (Model - Controler - View).

Modèle (model) gère la représentation des données : ce ne sont pas les données en elles-mêmes, mais une interface vers elles sans se préoccuper de comment elles sont gérées dans la base de données ;

Vue (view) gère quelles données sont présentées sur la page (fonction python) ;

Gabarit (template) gère comment les données sont présentées sur la page (html).

Le modèle MTV - schéma



Ici le contrôleur est le système en lui-même : la machinerie qui envoie une requête à la vue appropriée, selon la configuration d'URL de Django.

Projet Django et applications

Django distingue les projets des applications :

Application Application web qui gère une (ou des) tâche(s) en particulier (par exemple, une application de sondage) ;

Projet Ensemble de configurations et d'applications pour un site web donné.

Un projet peut contenir plusieurs applications, et une application peut être dans plusieurs projets.

II. Installer Django

Vérifier si Django est déjà présent

1. Taper la commande `python3` dans le terminal. Si cela ne marche pas, c'est qu'il faut installer Python [1].
2. Taper `import django`. Si cela ne marche pas, c'est qu'il faut installer Django.

Django inclut automatiquement le SGBD SQLite dont nous nous contenterons pour le séminaire, nous n'installerons donc pas de SGBD supplémentaire.

1. `https://www.python.org/downloads/`

Installer Django

1. Installer pip[2]. Si vous avez une version Python postérieure à la 3.4, il y est déjà, vous pouvez le mettre à jour avec
   `pip install --upgrade pip`
2. (Facultatif mais recommandé) Créer et activer un environnement virtuel pour ne pas installer Django sur tout le système
   1. Créer l'endroit où l'on souhaite mettre l'environnement virtuel. Par exemple, dans un dossier `.virtualenvs/` à la racine.
   2. Le créer avec `python3 -m venv ~/.virtualenvs/django` (ici, `django` est le nom de l'environnement virtuel)
   3. L'activer par
      `source ~/.virtualenvs/django/bin/activate`
3. (Après avoir activé l'environnement virtuel),
   `pip install Django`

2. `pip.pypa.io/en/latest/installing/#installing-with-get-pip-py`

Utiliser Django

- Sans environnement virtuel, il n'y a rien à faire ;
- Avec, il suffit de taper
  `source ~/.virtualenvs/python/bin/activate`.

III. Réaliser un projet en Django

Création du projet

Créons le projet `mysite` avec :
`django-admin startproject mysite`.

On obtient alors :

```
mysite/
    manage.py
    mysite/
        __init__.py
        settings.py
        urls.py
        wsgi.py
```

- Le `mysite/` racine est juste un dossier contenant le projet, qu'il est possible de renommer.
- `manage.py` permet de lancer le serveur, faire less tests, ...
- Le sous-répertoire `mysite/` correspond au paquet Python effectif de votre projet. C'est le nom à utiliser pour importer ce qu'il contient.

Lancer le serveur

Utiliser la commande : `python3 manage.py runserver`
Attention : ce serveur est fait pour tester pendant le développement, mais n'est pas adapté à un environnement de production !
Ensuite, on peut voir le résultat à l'adresse `http://127.0.0.1:8000`

Création d'une app

Créons une application nommée `polls` avec la commande suivante :
`python3 manage.py startapp polls`

On obtient le répertoire de notre application, qui est structuré de la façon suivante :

```
polls/
    __init__.py
    admin.py
    apps.py
    migrations/
        __init__.py
    models.py
    tests.py
    views.py
```

Création d'une view

Dans `polls/views.py` :

```
1 from django.http import HttpResponse
2
3 def index(request):
4     return HttpResponse("Here is the polls index.")
```

Pour appeler cette vue, il s'agit de l'associer à une URL, et pour cela nous avons besoin d'un URLconf (module de configuration d'URL).

Création d'un URLconf

Créez un fichier `polls/urls.py` et écrivez-y :

```
1 from django.urls import path
2 from . import views
3
4 urlpatterns = [
5     path('', views.index, name='index'),
6 ]
```

Pour le moment, `mysite` ne sait pas où aller chercher ce bout de code.

Création d'un URLconf - suite

Modifions `mysite/urls.py` :

```
1 from django.contrib import admin
2 from django.urls import include, path
3
4 urlpatterns = [
5     path('polls/', include('polls.urls')),
6     path('admin/', admin.site.urls),
7 ]
```

Quand Django rencontre un `include()`, il tronque le bout d'URL qui correspondait jusque là et passe la chaîne de caractères restante à la configuration d'URL incluse pour continuer le traitement.

Testons !

Lancez la commande `python3 manage.py runserver` puis rendez-vous à l'adresse `http://localhost:8000/polls/`.

Création de modèles

Dans `polls/models.py`, créons les modèles `Question` et `Choice`.

```
1  from django.db import models
2
3  class Question(models.Model):
4      q_text = models.CharField(max_length=200)
5      pub_date = models.DateTimeField('date
6          published')
7
8
9  class Choice(models.Model):
10     question = models.ForeignKey(Question, \
11         on_delete=models.CASCADE)
12     choice_text = models.CharField(max_length=200)
13     votes = models.IntegerField(default=0)
```

Création de modèles - remarques

- Chaque modèle est représenté par une classe qui hérite de `django.db.models.Model`.
- Nous ne créons pas nous-mêmes les champs dans la base de données, c'est Django qui s'en charge ! Dans notre code, on peut les appeler par les noms que nous avons choisis (qui seront des noms de colonne).
- Ici, on définit une relation avec `ForeignKey`. Cela indique à Django que chaque `Choice` n'est relié qu'à une seule `Question`. Il est également possible de faire des relations plusieurs-à-un ou plusieurs-à-plusieurs.
- Django ajoute automatiquement une clef primaire, pas besoin de le faire à la main !

Migrations

Pour inclure l'application dans notre projet, nous avons besoin d'ajouter une référence à sa classe de configuration dans le réglage `INSTALLED_APPS`, présent dans `mysite/settings.py`.

```
1 INSTALLED_APPS = [
2     'polls.apps.PollsConfig',
3     'django.contrib.admin',
4     'django.contrib.auth',
5     'django.contrib.contenttypes',
6     'django.contrib.sessions',
7     'django.contrib.messages',
8     'django.contrib.staticfiles',
9 ]
```

Migrations (2)

Appliquons les commandes suivantes :

1. `python3 manage.py makemigrations polls` pour indiquer à Django que nous avons effectué des changements aux modèles pour qu'ils soient stockés sous forme de migration ;
2. `python3 manage.py migrate` qui applique toutes les migrations (ie, modifie la base de données) des `INSTALLED_APPS` qui n'ont pas déjà été appliquées.

Migrations (résumé)

Modifier les modèles se fait en trois étapes :

1. Modifier `models.py` ;
2. `python3 manage.py makemigrations polls` ;
3. `python3 manage.py migrate`.

Petit interlude

Modifions nos modèles en ajoutant la méthode `__str__`, qui va être utile pour ce que nous allons voir ensuite (et qui est très utile en général).

```
1  from django.db import models
2
3  class Question(models.Model):
4      # ...
5      def __str__(self):
6          return self.q_text
7
8  class Choice(models.Model):
9      # ...
10     def __str__(self):
11         return self.choice_text
```

Site d'administration de django

1. Créons un·e superutilisateurice avec `python3 manage.py createsuperuser` ;
2. Lançons le serveur avec `python3 manage.py runserver` ;
3. Rendons nous à `http://127.0.0.1:8000/admin/`

Nous voyons quelques types de contenu éditable : groupes et utilisateurs. Ils sont fournis par django.contrib.auth, le système d'authentification livré avec Django. Mais nos modèles ne sont présents...

Rendre l'application de sondage modifiable via l'interface d'admin

Il faut indiquer à l'admin que les objets ont une interface d'administration. Modifions `polls/admin.py` à cet effet :

```
1 from django.contrib import admin
2 from .models import Question
3
4 admin.site.register(Question)
```

Ici on n'ajoute pas d'interface pour `Choice`, mais on pourrait le faire.
Retournons sur `http://127.0.0.1:8000/admin/`. On peut créer, supprimer, modifier des `Question` ! (On pourrait aussi les faire en passant par le shell python `python3 manage.py shell`).

Les tests

Les tests, c'est le bien. Cela permet de :

- Gagner du temps ;
- Vérifier plus rigoureusement, plus rapidement ;
- Gagner la confiance des autres ;
- Prévenir la casse.

On écrit les tests dans `polls/tests.py` (en définissant une classe `QuestionModelTests` (ou un autre nom) qui hérite de la classe `TestCase` et dont les méthodes commencent par `test_`) puis on les exécute avec `python3 manage.py test polls` (ou `python3 manage.py test` pour tester tout le projet).

IV. Conclusion

Le tutoriel django (disponible en français) `https://docs.djangoproject.com/en/2.1/intro/tutorial01/` est très bien fait !