Résumé du cours Django — Framework Python pour le Web (Django==4.2)

Chaima Fouzri

27 octobre 2025

1. Introduction à Django

Django en bref

Django est un **framework web Python** basé sur le modèle **MVC (Modèle–Vue–Contrôleur)**, ou plus précisément **MVT (Model–View–Template)**. Il permet de créer rapidement des applications web robustes, sécurisées et modulaires.

2. La couche modèle (Models Layer)

C'est la partie qui gère les **données** et la **base de données**.

2.1. Concepts abordés

- **Types de champs** : `CharField`, `TextField`, `IntegerField`, `DateTimeField`, `BooleanField`, `ForeignKey`, etc.
- **Options des champs** : `null`, `blank`, `default`, `choices`, `unique`.
- **Associations entre modèles** :
  - `OneToOneField` (relation 1–1)
  - `ForeignKey` (plusieurs–à–1)
  - `ManyToManyField` (plusieurs–à–plusieurs)
- **Gestion des migrations** :
  - `python manage.py makemigrations`
  - `python manage.py migrate`

2.2. Fonctions et classes importantes

— `__str__(self)` : personnalise l'affichage des objets.
— `class Meta:` : configure le comportement du modèle (ordre, nom, permissions...).

3. Les requêtes ORM

L'ORM (*Object Relational Mapper*) permet de manipuler la base sans écrire de SQL.

— **Querysets** : ensembles d'objets récupérés par l'ORM (`.filter()`, `.all()`, `.exclude()`...)
— **Lookups** : filtres avancés (`field__icontains`, `field__gte`, `date__range`)
— **Classe Q** : requêtes complexes avec des **OU** / **ET** logiques.

```
from django.db.models import Q
User.objects.filter(Q(age__gte=18) | Q(is_staff=True))
```

— **Validators** : contraintes personnalisées sur les champs (email, longueur minimale, etc.)

4. Génération de la base de données

Processus

— Django crée la base à partir des modèles et migrations.
— Teste la structure avant SQL.
— Étude de cas : modèle complet avec relations, puis génération de la BD.

5. Administration Django

Permet de gérer les données via une **interface graphique** sans coder.

```
list_display = ('nom', 'date_creation')
search_fields = ('nom',)
list_filter = ('categorie',)
```

— **InlineModelAdmin** : gère les relations directement dans une page admin.
— **Actions personnalisées** : ex. "Marquer comme validé".

6. Gestion des utilisateurs et authentification

— Django possède un système d'authentification intégré.

— Gestion :
  — Utilisateurs (`User`)
  — Groupes et permissions
  — Connexion / déconnexion
  — Profils personnalisés via `AbstractUser` ou `OneToOneField`

7. Dashboard Admin (Étude de cas)

— Création d'un tableau de bord personnalisé :
  — Statistiques, filtres et actions rapides
  — Templates HTML + Bootstrap pour une meilleure apparence

8. Workshop IA

Bonus IA

— Intégration d'Intelligence Artificielle (analyse audio, reconnaissance vocale, émotions).

— Objectif : intégrer une IA dans un projet Django (Whisper, NLP...).

9. Déploiement sur Render

— Déployer sur **Render.com** :
  — Création du repo GitHub
  — Fichier `requirements.txt`
  — Configuration du service Render
  — Base PostgreSQL et variables d'environnement

10. Résumé global

Le cours enseigne à :

1. Créer un projet Django complet (modèles, vues, templates).
2. Utiliser l'ORM et gérer les migrations.
3. Gérer l'authentification et le tableau admin.
4. Intégrer des modules IA.
5. Déployer en ligne sur Render.

Fin du résumé — Framework Django 4.2

Préparé par Chaima Fouzri — Esprit 5TWIN4