

## TP03 Python Django – Shell et site d'administration

fait par :

KOUTEMA Ditoma

## **Table des matières**

<b>1</b>	<b>Mise en route</b>	<b>3</b>
<b>2</b>	<b>L'interface de programmation (API)</b>	<b>3</b>
<b>3</b>	<b>Le site d'administration</b>	<b>7</b>

## 1 Mise en route

Pour travailler dans les bonnes conditions, je suis rentré dans mon projet et j'ai fait un clic droit et en suite j'ai ouvert le terminal a partir de la. Puis j'ai lancer le serveur de développement comme suit : [py manage.py runserver](#)

## 2 L'interface de programmation (API)

1.

- **API** : Application Programming Interface.
- L'api permet a deux applications distinctes des communiquer et d'échanger des données.
- Un shell est un interpréteur de commandes destiné aux systèmes d'exploitation de type linux qui permet d'accéder aux fonctionnalités internes du système d'exploitation.

2.

- Quand on lance la commande : [py manage.py shell](#), il ouvre le shell de python avec sa version installé.

```
Python 3.10.12 (main, Jun 11 2023, 05:26:28) [GCC 11.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
```

```
>>>
```

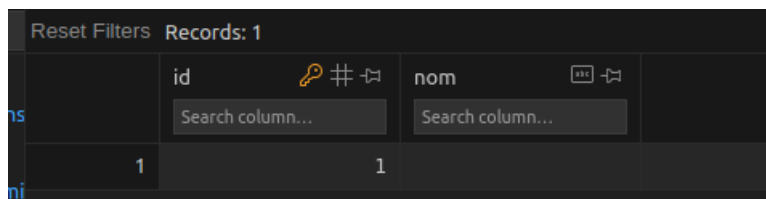
- Quand je lance la commande alias seule : [py](#)

```
Python 3.10.12 (main, Jun 11 2023, 05:26:28) [GCC 11.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

- Donc la différence entre [py manage.py shell](#) et [py](#) : Avec la première commande il y a l'information ([InteractiveConsole](#)) qui s'ajoute dans le shell python. Le fichier utilisé dans le premier cas est [manage.py](#)

3.

- a) Pour voir ce que contient l'attribut je me suis servie de l'extension de visual studio code qui s'appelle [sqlite viewer](#). Ici l'attribut ne contient rien.



The screenshot shows the SQLite Viewer interface. At the top, it says 'Reset Filters' and 'Records: 1'. Below this is a table with two columns: 'id' and 'nom'. Each column has a search bar labeled 'Search column...'. The table contains one row with the value '1' in the 'id' column and an empty cell in the 'nom' column.

id	nom
1	

- Oui c'est normal.
- Il n'y a pas d'erreur car il est unique ce qui veut dire qu'il peut donc être [null](#)

b) Création du niveau2 :

```
>>> from notes.models import *

>>> niveau2 = Niveau()
>>> niveau2.save()
```

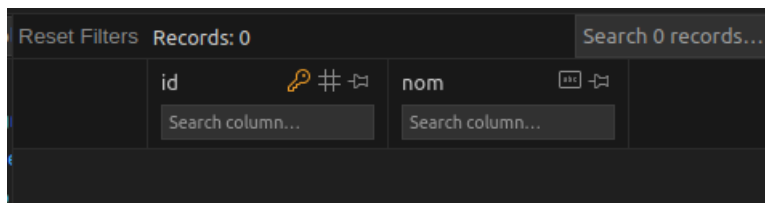
- Après la sauvegarde de l'objet niveau2 il y a une erreur qui s'est produite.
- Cette erreur est due à la violation de la contrainte unique.

c) Suppression du niveau sauvegardé dans la base de donnée :

- On doit sélectionner l'objet d'abord et en suite le supprimer.

```
>>> niveau = Niveau.objects.get(pk=1)
>>> niveau.delete()
(1, {'notes.Niveau': 1})
>>>
```

- Pour preuve je vais retourner sur visual studio code et me servir de sq-lite. viewer.



d) Création des niveaux :

```
>>>
>>> niveau1 = Niveau(nom='L1')
>>> niveau1.save()
>>> niveau2 = Niveau(nom='L2')
>>> niveau2.save()
>>> niveau3 = Niveau(nom='L3')
>>> niveau3.save()
>>>
```

- Valeurs des clés primaire : 2, 3, 4

Reset Filters Records: 3

	id	nom	
	<input data-bbox="491 383 671 412" type="text" value="Search column..."/>	<input data-bbox="699 383 879 412" type="text" value="Search column..."/>	
1	2	L1	
2	3	L2	
3	4	L3	

e) Création de 4 Élevés :

```
>>> eleve1 = Eleve(nom='Toune', prenom='Ouvo', sexe='M', date_naissance='1995-10-25', id='90454846', niveau=niveau1)
>>> eleve1.save()
>>> eleve2 = Eleve(nom='Falle', prenom='Éppa', sexe='F', date_naissance='1993-5-2', id='90154234', niveau=niveau2)
>>> eleve2.save()
>>> eleve3 = Eleve(nom='Jean', prenom='Almar', sexe='M', date_naissance='1993-12-6', id='90153598', niveau=niveau3)
>>> eleve3.save()
>>> eleve4 = Eleve(nom='Tartaglia', prenom='Rinascimento', sexe='M', date_naissance='1983-12-6', id='901538', niveau=niveau3)
>>> eleve4.save()
>>>
```

f) Créations de trois enseignants.

```
>>> ens1 = Enseignant(nom='Claude', prenom='Stroffaube', sexe='M', date_naissance='1967-8-1')
>>> ens1.save()
>>> ens2 = Enseignant(nom='Abla', prenom='Sillon', sexe='F', date_naissance='1960-7-2')
>>> ens2.save()
>>> ens3 = Enseignant(nom='Parlaf', prenom='Eunaitre', sexe='M', date_naissance='1990-2-28')
>>> ens3.save()
>>>
```

g) Créations de 5 matières.

```
>>>
>>> mat1 = Matiere(nom='Base de la programmation', enseignant=ens1)
>>> mat1.save()
>>> mat1.niveaus.add(niveau1)
>>>
>>> mat2 = Matiere(nom='Mathematiques', enseignant=ens2)
>>> mat2.save()
>>> mat2.niveaus.add(niveau1, niveau2)
>>>
>>> mat3 = Matiere(nom='Langages Web', enseignant=ens2)
>>> mat3.save()
>>> mat3.niveaus.add(niveau2, niveau3)
>>>
>>> mat4 = Matiere(nom='Gestion de projets', enseignant=ens1)
>>> mat4.save()
>>> mat4.niveaus.add(niveau3)
>>>
>>> mat5 = Matiere(nom='Anglais', enseignant=ens3)
>>> mat5.save()
>>> mat5.niveaus.add(niveau1, niveau2, niveau3)
>>>
```

4.

- Affichons l'objet Élevé correspondant a 'Jean'.

```
>>> print(Eleve.objects.get(nom='Jean'))
Eleve object (90153598)
>>>
```

- Ce qui s'affiche c'est la classe Élevé, son type et sa clé primaire. Voilà donc comment voir sa .
- Oui se serait mieux. Donc pour le faire, je crée la méthode spéciale `__str__` dans mon modèle qui retourne le nom et le prénom comme suit :

```
def __str__(self):
    return self.nom + " " + self.prenom
```

- Code

```
from django.db import models
```

```
# Create your models
here.
```

```
class Personne(models.Model):
    nom = models.CharField(max_length=50)
    prenom = models.CharField(max_length=50)
    sexe = models.CharField(max_length=100,
        choices= (('M', 'Masculin'), ('F', 'Feminin')) )
    date_naissance = models.DateField()
```

```
class Meta:
    abstract = True
```

```
class Niveau(models.Model):
    nom = models.CharField(max_length=2, unique=True)
```

```
def __str__(self):
    return self.nom
```

```
class Enseignant(Personne):
    def __str__(self):
        return self.nom + " " + self.prenom + " " + self.sexe +
            " " + self.date_naissance
```

```
class Matiere(models.Model):
```

```
def __str__(self):
    return self.nom + " " + self.enseignant
```

```
class Note(models.Model):
    eleve = models.ForeignKey(Eleve, on_delete=models.CASCADE)
    valeur = models.FloatField(max_length=30, null=True)

    def __str__(self):
        return self.eleve + " " + self.valeur
```

### 3 Le site d'administration

- 7

ting.py du projet et modifier la ligne suivante : `LANGUAGE_CODE = 'fr-FR'`

Fixons les attributs qui seront visible par l'utilisateur

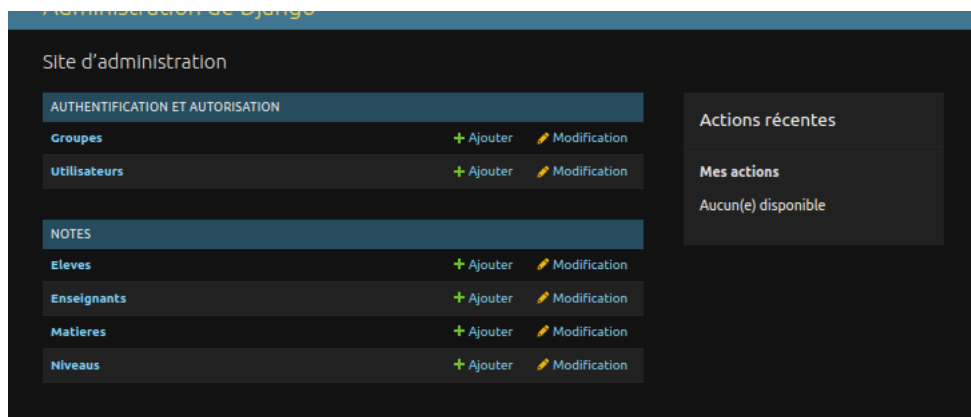
3. • Le fichier `notes/admin.py` nous permet d'enregistrer nos tables.
- La ligne qu'il faut ajouter est une importation des modèles à utiliser.

```
from notes.models import Eleve, Enseignant, Matiere, Niveau
```

Report du code :

```
from django.contrib import admin
from notes.models import Eleve, Enseignant, Matiere, Niveau
# Register your models here.
```

```
admin.site.register(Niveau)
admin.site.register(Eleve)
admin.site.register(Enseignant)
admin.site.register(Matiere)
Ce qu'on obtient comme résultat :
```



4. — Oui il y a une erreur au niveau du modèle Élève, car il n'y pas d'accent.

— Pour remédier a ce problème, on ajoute ce morceau de code suivant :

```
class Meta:
    verbose_name_plural = 'Élève'
```