

Institut Supérieur Numérique

Cours Développement web avec python

Le Framework Django

Présenté par : Sidi Mohamed Hmade

Présentation de django

❖ Historique :

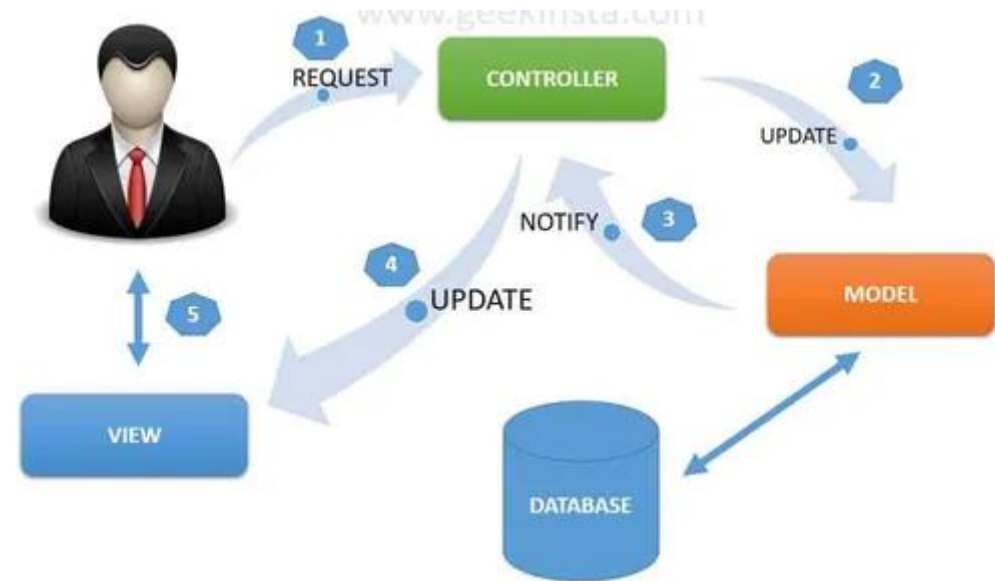
- ✓ Développé à l'origine à partir de 2003 pour un journal local de la ville de Lawrence.
- ✓ But : Réaliser une sorte de CMS, simple à utiliser pour les non informaticiens.
- ✓ En Open Source sous Licence depuis 2005.

❖ Django c'est quoi?

- ✓ Django est un framework python open-source consacré au développement web.
- ✓ Django s'inspire du modèle MVC (disons plutôt MVT), c'est-à-dire que la structure du framework sépare les données (models) qui sont séparées des traitements (controller) qui sont eux-mêmes séparés de la vue (view / template).
- ✓ Django, avec ORM, facilite la communication avec la base de données, sans avoir à écrire instructions SQL complexes.

MVC

MVC : Model-View-Controller : sépare le code en trois composants, l'architecture MVC est devenue populaire pour la conception d'applications Web ainsi que d'applications mobiles.



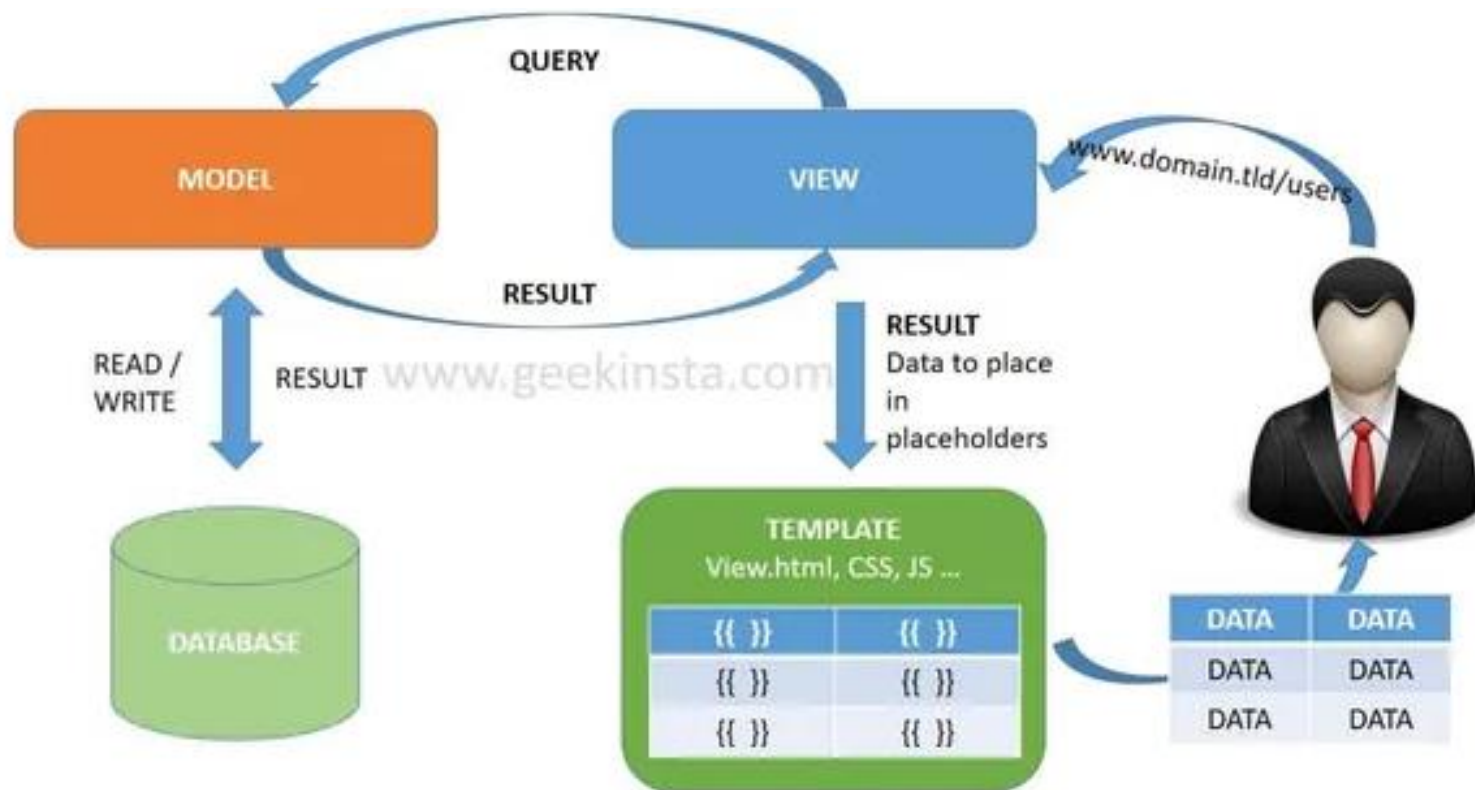
Modèle : Cette couche traite de la logique liée aux données. Par exemple, il peut récupérer, modifier et enregistrer des données dans la base de données.

Vue : c'est la couche de présentation. Dans une application Web, tout ce qui est affiché dans le navigateur relève de la vue.

Contrôleur : Il contrôle l'interaction entre la vue et le modèle. Par exemple, un contrôleur, en fonction d'une demande ou d'une action, collectera des données à partir d'une base de données à l'aide de Model et les enverra à l'utilisateur via Vue.

MVT

MVT : Model-View-Template est un autre modèle de conception similaire à MVC. Comme MVC, le modèle de conception MVT sépare également le code en trois parties.



MVT

- ❖ **Modèle** : Identique au modèle dans MVC. Il contient le code responsable du traitement des données et des bases de données.
- ❖ **Vue** : Dans le modèle de conception MVT, la vue décide quelles données doivent être affichées, gestionnaire de demandes qui renvoie le Template et le contenu appropriés, en fonction de la demande de l'utilisateur.
- ❖ **Template**: Il définit la façon dont les données sont présentées.

Installation de Django

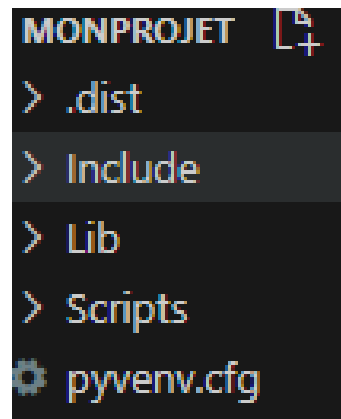
- ❖ L'installation de Python via l'installateur officiel, nous allons prendre la version 3



- ❖ Django est installé en utilisant pip, avec cette commande: Windows :
 - `pip install Django`
- ❖ Vous pouvez vérifier si Django est installé en demandant son numéro de version comme ceci :
 - `django-admin --version`

Créer un environnement virtuel

- ❖ Vous pouvez créer un environnement virtuel en tapant ceci dans l'invite de commande : **python -m venv monprojet**
- ❖ Cela permettra de configurer un environnement virtuel et de créer un dossier nommé « monprojet » avec des sous-dossiers et des fichiers, comme ceci :

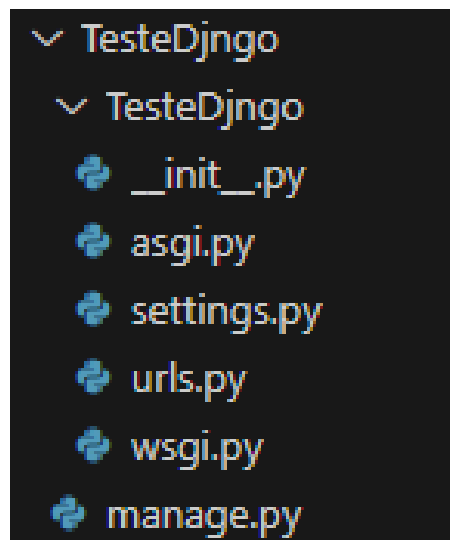


Créer un environnement virtuel

- ❖ Ensuite, vous devez activer l'environnement, en tapant cette commande :
 - `monprojet\Scripts\activate.bat`
- ❖ Vous devez activer l'environnement virtuel chaque fois que vous ouvrez l'invite de commande pour travailler sur votre projet.
- ❖ Enfin, nous pouvons installer Django :
 - `python -m pip install Django`
- ❖ Vous pouvez désactiver un environnement virtuel à tout moment en exécutant la commande `deactivate` dans l'invite de commandes. Cela revient à revenir à l'environnement Python global.

Créer un projet Django

- ❖ Naviguez jusqu'à l'endroit du fichier système dans lequel vous souhaitez stocker le code(`cd monprojet`) et exécutez cette commande dans l'invite de commande :
 - `django-admin startproject TesteDjngo`
- ❖ Django crée le dossier TesteDjngo avec ce contenu :



Exécuter le projet Django

- ❖ Accédez au dossier TesteDjngo et exécutez cette commande :
 - `python manage.py runserver`

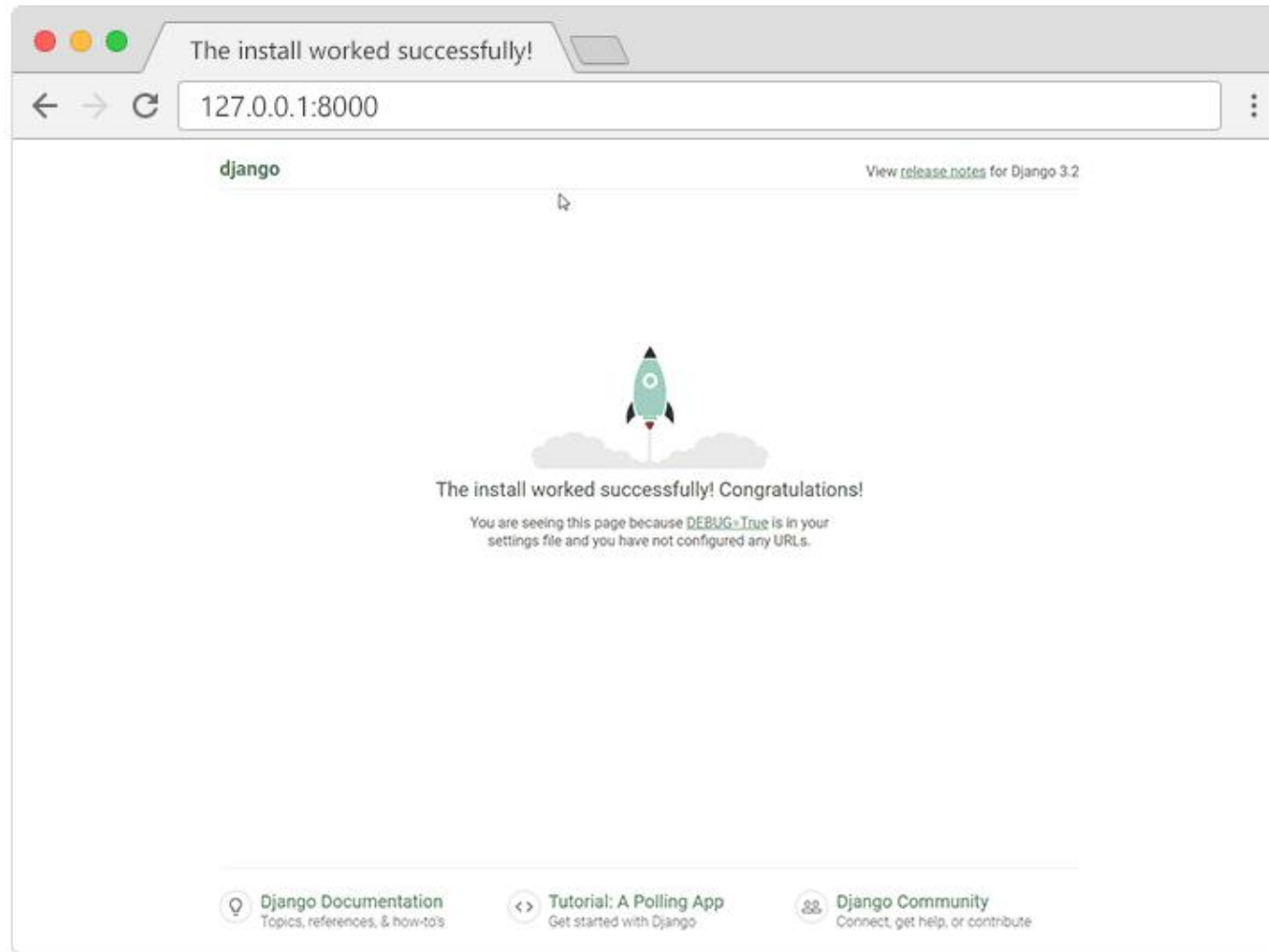
- ❖ Ce qui produira ce résultat :

```
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced)
You have 18 unapplied migration(s). Your project may not work properly until you apply the migrations
Run 'python manage.py migrate' to apply them.
November 24, 2022 - 16:32:22
Django version 4.1.3, using settings 'myworld.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
[24/Nov/2022 16:32:36] "GET / HTTP/1.1" 200 10681
[24/Nov/2022 16:32:36] "GET /static/admin/css/fonts.css HTTP/1.1" 200 423
[24/Nov/2022 16:32:36] "GET /static/admin/fonts/Roboto-Regular-webfont.woff HTTP/1.1" 200 85876
[24/Nov/2022 16:32:36] "GET /static/admin/fonts/Roboto-Light-webfont.woff HTTP/1.1" 200 85692
[24/Nov/2022 16:32:36] "GET /static/admin/fonts/Roboto-Bold-webfont.woff HTTP/1.1" 200 86184
Not Found: /favicon.ico
[24/Nov/2022 16:32:36] "GET /favicon.ico HTTP/1.1" 404 2111
```

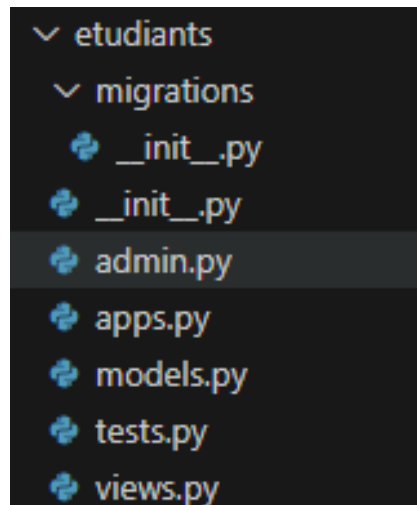
Exécuter le projet Django

- ❖ Ouvrez une nouvelle fenêtre de navigateur et tapez d'adresse 127.0.0.1:8000
- ❖ Le résultat :



Créer une application Django

- ❖ Créons l'application Django «etudiants» simple qui affiche « Hello word! ».
- ❖ Si le serveur est toujours en cours d'exécution et que vous ne parvenez pas à écrire des commandes, appuyez sur [CTRL] [PAUSE] pour arrêter le serveur.
- ❖ Lancer la commande :
 - `python manage.py startapp etudiants`
- ❖ Django crée un dossier etudiants dans monprojet, avec ce contenu



Les vues Django

- ❖ Les vues créent un contexte, intègre un template puis retournent un objet de type `HttpResponse`.
- ❖ Les vues sont généralement placées dans un fichier appelé `views.py`
- ❖ Il y a le fichier `views.py` dans votre dossier `etudiants` qui ressemble à ceci.
- ❖ Trouvez `etudiants/views.py` et remplacez son contenu par ceci :

```
from django.http import HttpResponse  
  
def index(request):  
    return HttpResponse("Hello world!")
```

URL

- ❖ Créez un fichier nommé `urls.py` dans le même dossier « `etudiants` » et tapez le code suivant :

```
from django.urls import path
from . import views

urlpatterns = [
    path("", views.index, name='index'), ]
```

- ❖ Le fichier `etudiants\urls.py` crée est spécifique à l'application.

URL

- ❖ Pour effectuer également un routage dans le répertoire racine, trouver le fichier [Testdjango/urls.py](#) et tapez :

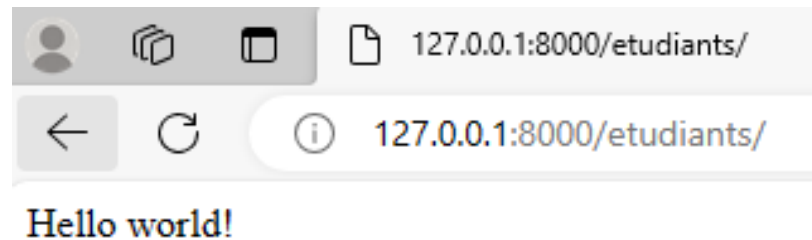
```
from django.contrib import admin  
from django.urls import include, path
```

```
urlpatterns = [  
    path('etudiants/', include('etudiants.urls')),  
    path('admin/', admin.site.urls),]
```

- ❖ accédez au dossier :/Testedjango et exécutez cette commande dans l'invite de commande :
 - python manage.py runserver

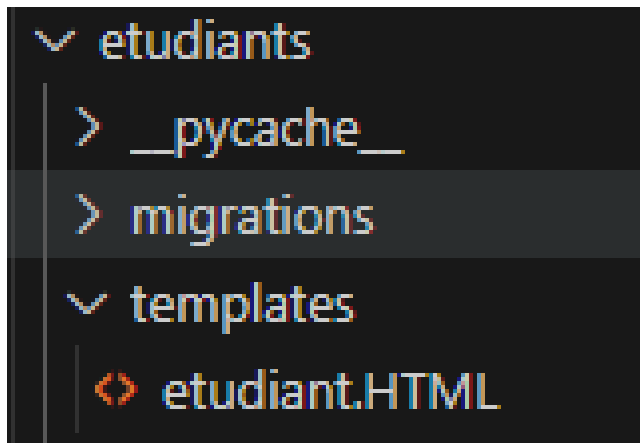
URL

- ❖ Dans le navigateur, tapez dans la barre d'adresse : 127.0.0.1: 8000/etudiants/
- ❖ Résultat :



Template Django

- ❖ L'affichage devrait être en HTML, et il devrait être créé dans un Template.
- ❖ Créer un dossier nommé templates à l'intérieur du dossier etudiant et créer un fichier HTML etudiant.html.
- ❖ La structure du fichier doit ressembler à ceci :



Template Django

❖ Ouvrez le fichier HTML et insérez ce qui suit :

```
<!DOCTYPE html>  
<html>  
<body>  
  
<h1>Bienvenu </h1>  
<p>Mon premier projet Django!</p>  
  
</body>  
</html>
```

Template Django

❖ Modifier la vue :

Ouvrez le fichier `etudiants/views.py` et remplacez la vue d'index par ceci :

```
from django.http import HttpResponse
from django.template import loader

def index(request):
    template = loader.get_template('etudiant.html')
    return HttpResponse(template.render())
```

Template Django

❖ Modifier les paramètres:

- ✓ Pouvoir travailler avec des trucs plus compliqués que « Hello World! », Nous devons dire à Django qu'une nouvelle application est créée.
- ✓ Cela se fait dans le fichier `TesteDjngo/settings.py`.
- ✓ Trouvez la liste `INSTALLED_APPS[]` et ajoutez l'application `etudiants` comme ceci

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'etudiants.apps.EtudiantsConfig'  
]
```

Template Django

❖ Exécutez ensuite cette commande :

✓ `python manage.py migrate`

❖ Ce qui produira ce résultat:

```
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... 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 auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying sessions.0001_initial... OK
```

Template Django

- ❖ Démarrez le serveur en accédant au dossier /TesteDjngo et exécutez cette commande.
 - ✓ `python manage.py runserver`
- ❖ Dans la fenêtre du navigateur, tapez dans la barre d'adresse `127.0.0.1:8000/etudiants/`
- ❖ Le résultat devrait ressembler à ceci:



Modèles Django

- ❖ Modèles : c'est la couche d'accès aux données.
- ❖ Base de données SQLite : Lorsque nous avons créé le projet Django TesteDjngo, nous avons obtenu une base de données SQLite vide.
- ❖ Créer une table (**modèle**) Pour créer une nouvelle table, nous devons créer un nouveau modèle.
 - ✓ Ouvrez le fichier /etudiants/models.py
 - ✓ Il est presque vide par défaut, avec seulement une instruction import et un commentaire.

Modèles Django

❖ members/models.py :

```
TesteDjngo > etudiants > + models.py
1  from django.db import models
2
3
```

❖ Pour ajouter une table Etudiants dans notre base de données, commencez par créer une classe Etudiants, et décrivez les champs de table qu'il contient :

```
from django.db import models
class Etudiants(models.Model):
    nom = models.CharField(max_length=255)
    prenom = models.CharField(max_length=255)
```

Modèles Django

❖ Accédez ensuite au dossier /TesteDjngo/ et exécutez cette commande

✓ `python manage.py makemigrations etudiants`

❖ Le résultat:

```
Migrations for 'etudiants':  
  etudiants\migrations\0001_initial.py  
    - Create model Etudiants
```

❖ Django crée un fichier avec toutes les nouvelles modifications et stocke le fichier dans le dossier [/migrations/](#)

❖ La prochaine fois que vous exécuterez `python manage.py migrate`, Django créera et exécutera une instruction SQL, basée sur le contenu de la nouvelle Migration.

Modèles Django

- ❖ Exécutez la commande migrate :
 - ✓ `python manage.py migrate`

- ❖ Le résultat :

```
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, etudiants, sessions
Running migrations:
  Applying etudiants.0001_initial... OK
```

- ❖ L'instruction SQL créée à partir du modèle est la suivante :

```
CREATE TABLE "etudiants_etudiants" (
  "id" INT NOT NULL PRIMARY KEY AUTOINCREMENT,
  "nom" varchar(255) NOT NULL,
  "prenom" varchar(255) NOT NULL);
```

- ❖ Vous avez maintenant une table Etudiants dans votre base de données !

Insérer des données

- ❖ La table des etudiants est vide.
- ❖ Dans la partie suivante, vous devez créer une interface utilisateur qui s'occuper des opérations CRUD (Créer, Lire, Mettre à jour, Supprimer), mais pour l'instant, on va écrire du code Python directement dans l'interpréteur Python (shell Python) et ajouter du code Python etudiants de notre base de données, sans l'interface utilisateur.
- ❖ Pour ouvrir un shell Python, tapez la commande suivante :
 - ✓ `python manage.py shell`
- ❖ le résultat devrait être quelque chose comme ceci :

```
Python 3.9.2 (tags/v3.9.2:1a79785, Feb 19 2021, 13:44:55) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>>
```

Insérer des données

- ❖ En bas, après les trois `>>>`, écrivez ce qui suit:
 - ✓ `from etudiants.models import Etudiants`
- ❖ Visualiser la table Etudiants :
 - ✓ `Etudiants.objects.all()`

Cela devrait vous donner un objet QuerySet vide, comme ceci :

```
<QuerySet []>
>>> from etudiants.models import Etudiants
>>> Etudiants.objects.all()
<QuerySet []>
```

- ❖ Un QuerySet est une collection de données provenant d'une base de données.

Insérer des données

- ❖ Ajoutez un enregistrement à la table en exécutant ces deux lignes :
 - ✓ `etudiant = Etudiants(nom='Med', prenom='Ali')`
 - ✓ `etudiant.save()`
- ❖ Exécutez cette commande pour voir si la table Etudiants a un étudiant :
 - ✓ `Etudiants.objects.all().values()`
- ❖ le résultat ressemblera à ceci :

```
>>> etudiant = Etudiants(nom = 'Med', prenom='Ali')
>>> etudiant.save()
>>> Etudiants.objects.all().values()
<QuerySet [{'id': 1, 'nom': 'Med', 'prenom': 'Ali'}]>
```

Ajouter plusieurs enregistrements

- ❖ Vous pouvez ajouter plusieurs enregistrements en faisant une liste d'objets, et exécuter sur chaque entrée : `Etudiants.save()`.

- ✓ `e1 = Etudiants(nom='Mariem', prenom='Ahmed')`
- ✓ `e2 = Etudiants(nom='Med', prenom='Ahmed')`
- ✓ `e3 = Etudiants(nom='Ahmed', prenom='Ali')`
- ✓ `e4 = Etudiants(nom='Ali', prenom='BA')`
- ✓ `e5 = Etudiants(nom='Tourad', prenom='Cheikh')`
- ✓ `etudiants_list = [e1, e2, e3, e4, e5]`

- ❖ Sauvegarder les données :

```
for x in etudiants_list :  
    x.save()
```

Ajouter plusieurs enregistrements

❖ Il y a maintenant 6 étudiants dans la table des étudiants :

✓ `Etudiants.objects.all().values()`

```
>>> Etudiants.objects.all().values()
<QuerySet [{'id': 1, 'nom': 'Med', 'prenom': 'Ali'}, {'id': 2, 'nom': 'Mariem',
'prenom': 'Ahmed'}, {'id': 3, 'nom': 'Med', 'prenom': 'Ahmed'}, {'id': 4, 'nom':
'Ahmed', 'prenom': 'Ali'}, {'id': 5, 'nom': 'Ali', 'prenom': 'BA'}, {'id': 6,
'nom': 'Tourad', 'prenom': 'Cheikh'}]>
```

Afficher la données

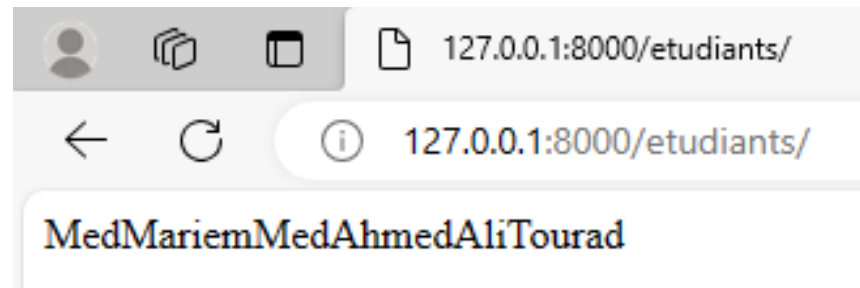
- ❖ Nous voulons voir le résultat dans une page Web, pas dans un environnement shell Python.
- ❖ Modifiez le contenu du fichier `etudiants/views.py` :

```
from django.http import HttpResponseRedirect
from django.template import loader
from .models import Etudiants

def index(request):
    MesEtds = Etudiants.objects.all().values()
    output = ""
    for x in MesEtds :
        output += x["nom"]
    return HttpResponseRedirect(output)
```

Afficher les données

- ❖ Si vous êtes toujours dans le shell Python, écrivez Cette commande pour quitter le Shell :
 - ✓ `quit()`
- ❖ Accédez au dossier `/MonProjet/` et tapez ceci pour démarrer le serveur
 - ✓ `python manage.py runserver`
- ❖ Dans la fenêtre du navigateur, tapez `127.0.0.1:8000/etudiants/` dans la barre d'adresse.
- ❖ Le résultat :



Ajouter un Template

- ❖ Dans le dossier templates, créez un fichier nommé index.html, avec le contenu suivant

```
<h1>Les Etudiants</h1>

<table border="1">
{% for x in MesEtds %}
<tr>
    <td>{{ x.id }}</td>
    <td>{{ x.nom }}</td>
    <td>{{ x.prenom }}</td>
</tr>
{% endfor %}
</table>
```

Modifier la vue

- ❖ Modifiez `etudiants/views.py` pour inclure le Template index

```
from django.http import HttpResponseRedirect
from django.template import loader
from .models import Etudiants
```

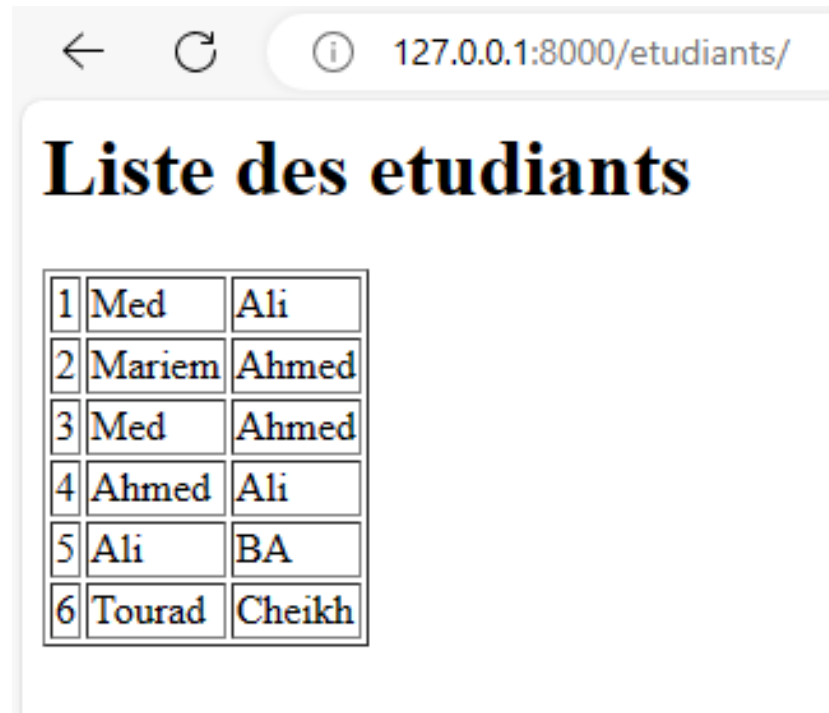
```
def index(request):
    MesEtds = Etudiants.objects.all().values()
    template = loader.get_template('index.html')
    context = { 'MesEtds': MesEtds,
    }
    return HttpResponseRedirect(template.render(context, request))
```

La vue index

- ❖ La vue index effectue les opérations suivantes :
 - ✓ Crée un objet MesEtds avec toutes les valeurs du modèle Etudiants.
 - ✓ Charge le Template index.html.
 - ✓ Crée un objet contenant l'objet MesEtds.
 - ✓ Envoie l'objet au Templates.
 - ✓ Affiche le code HTML rendu par le modèle.

La vue index

- ❖ Dans la fenêtre du navigateur, tapez dans la barre d'adresse.127.0.0.1:8000/etudiants/
- ❖ Le résultat :



1	Med	Ali
2	Mariem	Ahmed
3	Med	Ahmed
4	Ahmed	Ali
5	Ali	BA
6	Tourad	Cheikh

Insérer une ligne

- ❖ Jusqu'à présent, nous avons créé une table Etudiants dans notre base de données, et nous avons inséré cinq enregistrements en écrivant du code dans le shell Python.
- ❖ Maintenant, nous voulons pouvoir créer de nouveaux membres à partir d'une page Web.
- ❖ Ajouter un lien dans le Template : `etudiants/templates/index.html`

Insérer une ligne

- ❖ Ajouter un lien dans le Template : etudiants/templates/index.html

```
<h1>Les Etudiants</h1>
```

```
<table border="1">
```

```
{% for x in MesEtds %}
```

```
<tr>
```

```
    <td>{{ x.id }}</td>
```

```
    <td>{{ x.nom }}</td>
```

```
    <td>{{ x.prenom }}</td>
```

```
</tr>
```

```
{% endfor %}
```

```
</table>
```

```
<p> <a href="add/">Ajouter un etudiant </a> </p>
```

Insérer une ligne

❖ Le résultat :

Members

1	Emil	Refsnes
2	Mariem	Ahmed
3	Med	Ahmed
4	Ahmed	Ali
5	Ali	BA
6	Tourad	Cheikh

[Add member](#)

Insérer une ligne

- ❖ Ajouter un nouveau Template : add.html

```
<h1>Ajouter un etudiant</h1>
<form action="addrecord/" method="post">
{% csrf_token %}
    Nom:<br>
    <input name=« nom">
    <br><br>
    Prenom:<br>
    <input name=« prenom">
    <br><br>
    <input type="submit" value="Submit">
</form>
```

- ❖ Pour protéger un formulaire il vous suffira t'intégrer le template tag **csrf_token** dans votre template:

Insérer une ligne

- ❖ Ajoutez une vue add dans le fichier, etudiants/views.py

```
from django.http import HttpResponseRedirect
from django.template import loader
from .models import Etudiants
```

```
def index(request):
    MesEtds = Etudiants.objects.all().values()
    template = loader.get_template('index.html')
    context = { 'MesEtds': MesEtds,
    }
    return HttpResponseRedirect(template.render(context, request))
```

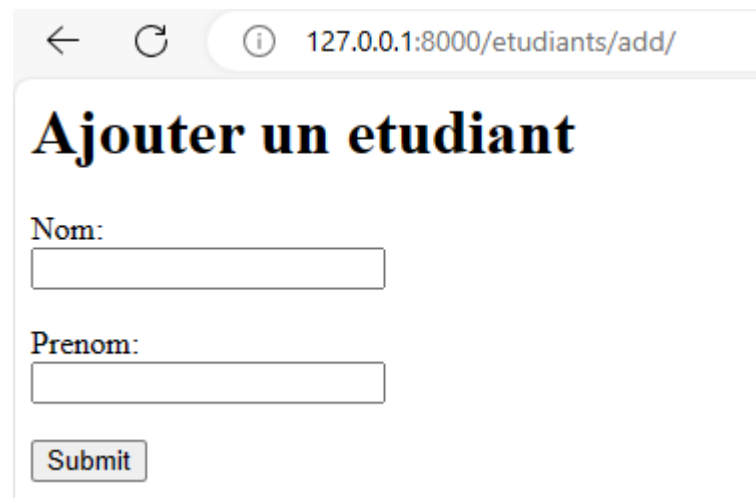
```
def add(request):
    template = loader.get_template('add.html')
    return HttpResponseRedirect(template.render({}, request))
```

Insérer une ligne

- ❖ Ajoutez une fonction path() dans le fichier, etudiants/urls.py

```
from django.urls import path
from . import views
urlpatterns = [
    path("", views.index, name='index'),
    path('add/', views.add, name='add'), ]
```

- ❖ Le résultat :



A screenshot of a web browser window. The address bar shows the URL '127.0.0.1:8000/etudiants/add/'. The page title is 'Ajouter un etudiant'. Below the title, there are two input fields: 'Nom:' and 'Prenom:'. At the bottom of the form is a 'Submit' button.

Insérer une ligne

- ❖ La vue addrecord pour insérer un etudiant.
- ❖ Assurez-vous d'ajouter la vue addrecord dans le fichier : etudiants/views.py

```
from django.http import HttpResponseRedirect
from django.template import loader
from django.urls import reverse
from .models import Etudiants
```

```
def addrecord(request):
    x = request.POST['nom']
    y = request.POST['prenom']

    etudiant = Etudiants(nom=x, prenom=y)
    etudiant.save()

    return HttpResponseRedirect(reverse('index'))
```

Insérer une ligne

- ❖ Nous devons donc ajouter une fonction `path()` dans le fichier `etudiants/urls.py`

```
from django.urls import path
from . import views
urlpatterns = [
    path('', views.index, name='index'),
    path('add/', views.add, name='add'),
    path('add/addrecord/', views.addrecord, name='addrecord'),
]
```

Insérer une ligne

❖ Tester l'insersion :

← ↻ ⓘ 127.0.0.1:8000/etudiants/add/

Ajouter un etudiant

Nom:

Prenom:

Les Etudiants

1	Med	Ali
2	Mariem	Ahmed
3	Med	Ahmed
4	Ahmed	Ali
5	Ali	BA
6	Tourad	Cheikh
7	Hacen	Ahmed

[Ajouter un etudiant](#)

Supprimer une ligne

- ❖ Pour supprimer un enregistrement, nous n'avons pas besoin d'un nouveau Template, mais nous devons apporter quelques modifications.
- ❖ Ajoutez une colonne « supprimer » dans le template étudiants/templates/index.html

```
<h1>Les etudiants</h1>
<table border="1">
    {% for x in MesEtds %}
    <tr> <td>{{ x.id }}</td>
    <td>{{ x.nom }}</td>
    <td>{{ x.prenom }}</td>
    <td><a href="delete/{{ x.id }}">delete</a></td>
    </tr>
    {% endfor %}
</table>
<p> <a href="add/">Ajouter un etudiant</a> </p>
```

Supprimer une ligne

❖ Le résultat :

Les etudiants

1	Med	Ali	delete
2	Mariem	Ahmed	delete
3	Med	Ahmed	delete
4	Ahmed	Ali	delete
5	Ali	BA	delete
6	Tourad	Cheikh	delete
7	Hacen	Ahmed	delete

[Ajouter un etudiant](#)

Supprimer une ligne

- ❖ Nous allons ajouter une URL dans le fichier `etudiants/urls.py`.

```
from django.urls import path
from . import views
urlpatterns = [
    path('', views.index, name='index'),
    path('add/', views.add, name='add'),
    path('add/addrecord/', views.addrecord, name='addrecord'),
    path('delete/<int:id>', views.delete, name='delete'),
]
```

- ❖ Maintenant, nous devons ajouter une nouvelle vue `delete` dans le fichier `etudiants/views.py`

```
def delete(request, id):
    e = Etudiants.objects.get(id=id)
    e.delete()
    return HttpResponseRedirect(reverse('index'))
```

Supprimer une ligne

❖ Tester :

← ↻ ⓘ 127.0.0.1:8000/etudiants/

Les etudiants

1	Med	Ali	delete
2	Mariem	Ahmed	delete
3	Med	Ahmed	delete
4	Ahmed	Ali	delete
5	Ali	BA	delete
6	Tourad	Cheikh	delete
7	Hacen	Ahmed	delete

[Ajouter un etudiant](#)

← ↻ ⓘ 127.0.0.1:8000/etudiants/

Les etudiants

1	Med	Ali	delete
2	Mariem	Ahmed	delete
3	Med	Ahmed	delete
4	Ahmed	Ali	delete
5	Ali	BA	delete
6	Tourad	Cheikh	delete

[Ajouter un etudiant](#)

Modifier une ligne

- ❖ Pour mettre à jour un enregistrement, nous avons besoin de l'ID de l'enregistrement, et nous avons besoin d'un template avec une interface Cela nous a permis de changer les valeurs.
- ❖ Nous devons d'abord apporter quelques modifications sur le template index.html

```
<h1>La liste des etudiants</h1>
<table border="1">
    {% for x in MesEtds %}
    <tr>
        <td><a href="update/{{ x.id }}">{{ x.id }}</a></td>
        <td>{{ x.nom }}</td>
        <td>{{ x.prenom }}</td>
        <td><a href="delete/{{ x.id }}">delete</a></td>
    </tr>
    {% endfor %}
</table>
<p> <a href="add/">Ajouter un etudiant</a> </p>
```

Modifier une ligne

❖ Le résultat :



← ↻ ⓘ 127.0.0.1:8000/etudiants/

La liste des etudiants

1	Med	Ali	delete
2	Mariem	Ahmed	delete
3	Med	Ahmed	delete
4	Ahmed	Ali	delete
5	Ali	BA	delete
6	Tourad	Cheikh	delete
8	Hacen	Ahmed	delete

[Ajouter un etudiant](#)

Modifier une ligne

❖ Ensuite, ajoutez la vue update dans le fichier etudiants/views.py

```
def update(request, id):  
    etud = Etudiants.objects.get(id=id)  
    template = loader.get_template('update.html')  
    context = { 'etud': etud, }  
    return HttpResponse(template.render(context, request))
```

Modifier une ligne

- ❖ Ajouter un nouveau Template : etudiants/templates/update.html
- ❖ Le Template contient un formulaire HTML avec les valeurs de l'étudiant sélectionné.

```
<h1>Modifier un etudiant</h1>
<form action="updaterecord/{{ etud.id }}" method="post">
    {% csrf_token %}
    Nom:<br>
    <input name="nom" value="{{ etud.nom }}">
    <br><br>
    Prenom:<br>
    <input name="prenom" value="{{ etud.prenom }}">
    <br><br>
    <input type="submit" value="Submit">
</form>
```

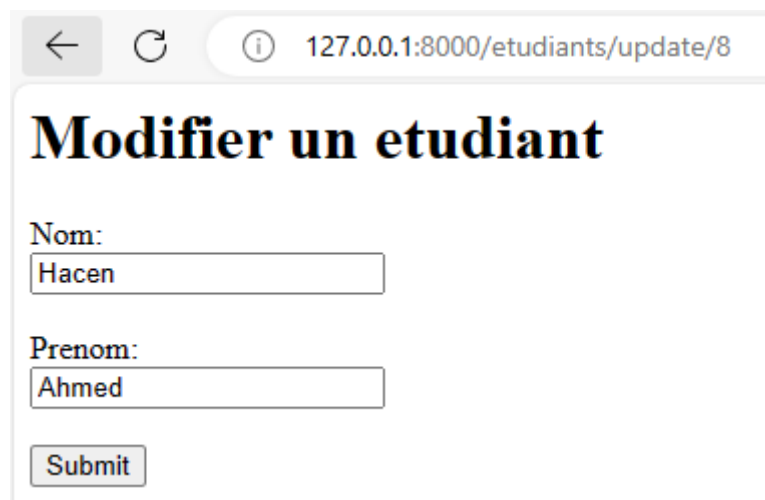
Modifier une ligne

- ❖ Ajouter une URL pour update dans members/urls.py

```
from django.urls import path
from . import views urlpatterns = [
    path('', views.index, name='index'),
    path('add/', views.add, name='add'),
    path('add/addrecord/', views.addrecord, name='addrecord'),
    path('delete/<int:id>', views.delete, name='delete'),
    path('update/<int:id>', views.update, name='update'),
]
```

Modifier une ligne

- ❖ Dans le navigateur, cliquez sur l'ID de l'étudiant que vous souhaitez modifier et le résultat devrait ressembler à ceci:



The screenshot shows a web browser window with the address bar displaying "127.0.0.1:8000/etudiants/update/8". The page title is "Modifier un etudiant". Below the title, there are two text input fields: "Nom:" with the value "Hacen" and "Prenom:" with the value "Ahmed". At the bottom of the form is a "Submit" button.

← ↻ ⓘ 127.0.0.1:8000/etudiants/update/8

Modifier un etudiant

Nom:

Prenom:

Modifier une ligne

❖ Nous devons donc ajouter une vue updaterecord dans etudiants/views.py

```
def updaterecord(request, id):  
    nom = request.POST['nom']  
    prenom = request.POST['prenom']  
    etd = Etudiants.objects.get(id=id)  
    etd.nom = nom  
    etd.prenom = prenom  
    etd.save()  
    return HttpResponseRedirect(reverse('index'))
```

Modifier une ligne

- ❖ Nous devons donc ajouter un path pour la vue updaterecord dans `etudiants/urls.py`

```
from django.urls import path
from . import views urlpatterns = [
    path('', views.index, name='index'),
    path('add/', views.add, name='add'),
    path('add/addrecord/', views.addrecord, name='addrecord'),
    path('delete/<int:id>', views.delete, name='delete'),
    path('update/<int:id>', views.update, name='update'),
    path('update/updaterecord/<int:id>', views.updaterecord,
name='updaterecord'),

]
```

Modifier une ligne

- ❖ Essayez de mettre à jour un enregistrement

La liste des etudiants

1	Med	Ali	delete
2	Mariem	Ahmed	delete
3	Med	Ahmed	delete
4	Ahmed	Ali	delete
5	Ali	BA	delete
6	Tourad	Cheikh	delete
8	Hacen	Ahmed	delete

[Ajouter un etudiant](#)

← ↻ ⓘ 127.0.0.1:8000/etudiants/update/8

Modifier un etudiant

Nom:

Prenom:

La liste des etudiants

1	Med	Ali	delete
2	Mariem	Ahmed	delete
3	Med	Ahmed	delete
4	Ahmed	Ali	delete
5	Ali	BA	delete
6	Tourad	Cheikh	delete
8	Hacen	Med	delete

[Ajouter un etudiant](#)