TP4 Python Django – Les vues

TOYI Francois

October 2024

Mise en route

Une **vue** est une fonction en Dango.
Nous aurons à travaillé beaucoup sur les fichiers vues dans se tp.

Ecriture de vues en code python.

1. l'URL de la racine de l'application est la suivante, on a :
http://127.0.0.1:8000/notes/.

2.b Il y à une erreur.

Pour contourner cet erreur on doit changer le niveau de notre import et le nom de notre module aussi comme suis, on a :

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

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

Vue eleve.py

```
from django.shortcuts import render
from django.http import HttpResponse


def eleve(request):
    return HttpResponse("Je suis dans ma vue eleve() !!")


def eleves(request,id):
    return HttpResponse("Je suis dans ma vue eleves !!")
```

Vue matiere.py

```
from django.shortcuts import render
from django.http import HttpResponse


def matiere(request):
    return HttpResponse("Je suis dans ma vue matiere")



def matieres(request,id):
    return HttpResponse("Je suis dans ma vue matieres")
```

Vue niveau.py

```
from django.shortcuts import render
from django.http import HttpResponse


def niveau(request,id):
    return HttpResponse("Je suis dans ma vue niveau()")
```

3. Pour que nos vues qu'on viens de créer soit visible on doit les ajouter où définir au niveau de notre fichier **urls.py**.

Voici le contenu du fichier **urls.py** :

```
 from django.urls import path
from .views import index, eleve, matiere, niveau

urlpatterns = [
    path('', index.index, name='index'),
    path('eleves/', eleve.eleves, name='eleves'),
    path('eleve/<int:id>/', eleve.eleve, name='eleve'),
    path('matieres/', matiere.matieres, name='matiere'),
    path('matiere/<int:id>', matiere.matiere, name='matiere'),
    path('niveau/<int:id>', niveau.niveau, name='niveau'),
]
```

4. Pour accéder aux pages de détail, on fait :

```
http://127.0.0.1:8000/notes/eleve
http://127.0.0.1:8000/notes/eleves/90158486/

http://127.0.0.1:8000/notes/matiere
http://127.0.0.1:8000/notes/matieres/1

http://127.0.0.1:8000/notes/niveau/1
```

5. Pour chacune des pages voici les modèles à utilisé, on a :

Pages eleves :

- Eleve
- Note

Pages matieres :

- Matiere

Pages niveau :

- Niveau

6. Voici le contenu de nos vue eleves et matieres, on a :

eleves :

```
def eleves(request):
    eleves_list = Eleve.objects.all()

    html_content = """
    <html>
        <head>
            <title>Liste des élèves et leurs notes</title>
        </head>
        <body>
            <h1>Liste des élèves et leurs notes</h1>
            <table border="1">
                <thead>
                    <tr>
                        <th>Nom de l'élève</th>
                        <th>Niveau</th>
                        <th>Matière</th>
                        <th>Note</th>
                    </tr>
                </thead>
                <tbody>
    """

    for eleve in eleves_list:
        for matiere in eleve.matieres.all():
            note = Note.objects.filter(eleve=eleve, matiere=matiere).first()
            note_value = note.valeur if note else 'Nulle'
            html_content += f"""
                <tr>
```

```
                    <td>{eleve.nom}</td>
                    <td>{eleve.niveau.nom}</td>
                    <td>{matiere.nom}</td>
                    <td>{note_value}</td>
                </tr>
            """

    # Fin du contenu HTML
    html_content += """
                </tbody>
            </table>
        </body>
    </html>
    """

    return HttpResponse(html_content)



Matiere :

def matieres(request):
    list_matiere = Matiere.objects.all()

    # Début du contenu HTML
    html_content = """
    <html>
        <head>
            <title>Liste des Matières</title>
        </head>
        <body>
            <h1>Liste des Matières</h1>
            <table border="1">
                <thead>
                    <tr>
                        <th>Nom de la Matière</th>
                    </tr>
                </thead>
                <tbody>
    """

    # Remplissage des matières dans la table
    for matiere in list_matiere:
        html_content += f"""
            <tr>
                <td>{matiere.nom}</td>
```

```
            </tr>
        """

    # Fin du contenu HTML
    html_content += """
                </tbody>
            </table>
        </body>
    </html>
    """

    return HttpResponse(html_content)
```

Templates

1. Un **templates** est un fichier qui contient du code html dans le quel ont peut ajouter la logique pour faire une affichage dynamique tout comme blade dans **laravel**.

2. ce code vas afficher dans le navigateur **Bonjour tout le monde !**

3. la méthode **render()** sert à générer une réponse HTTP en combinant un template HTML avec des données du contexte .

4. Comme vous nous aviez demandé plus haut de modifier le nom de notre fichier views.py en index.py on as :

Fichier index.py

```
from django.shortcuts import render
# from django.http import HttpResponse

# Create your views here.

def index(request):
 # return HttpResponse("Bonjour tout le monde !")
 return render(request,"notes/index.html")
```

Fichier eleves.html

```
    <h1>Liste des élèves avec leurs notes et moyennes</h1>
<ul>
    {% for eleve in eleves %}
    <!-- <li>{{ eleve }}</li> -->
    <li>
        <strong>{{ eleve.nom }}</strong> - Niveau: {{ eleve.niveau }}
```

```
        <ul>
            {% for matiere_note in eleve.matieres_notes %}
            <li>{{ matiere_note.matiere }} - Moyenne: {{ matiere_note.note }}</li>
            {% endfor %}
        </ul>
    </li>
    {% endfor %}
</ul>
```

Fichier matiere.html

```
    <h1>Liste des matières</h1>
<ul>
    {% for matiere in matieres %}
    <!-- <li>{{ matiere }}</li> -->
    <li>{{ matiere.matiere }} - Enseignant: {{ matiere.enseignants }}</li>
    {% endfor %}
</ul>
```

Gestion des erreurs

1. Ici on a pas encore mis en place de la méthode **render()** dans les autres vues, pour les restes des vues aussi l'une des particularitée est quels prennent en appel des paramètres(id) .

2. La fonction raccourci **get_object_or_404()** est une fonction fournie par Django qui permet de récupérer un objet depuis la base de données. Si l'objet correspondant n'est pas trouvé, elle lève automatiquement une exception **Http404**, qui renvoie une page **404** (page non trouvée) à l'utilisateur. Elle évite d'avoir à écrire manuellement un bloc **try-except** chaque fois qu'on souhaite gérer ce type d'erreur.

3. Ecrivons les vues et les templates restantes .

Vue eleve()

```
def eleve(request, id):
    detail_eleve = get_object_or_404(Eleve, id=id)

    matieres_notes = []
    for matiere in detail_eleve.matieres.all():
        note = Note.objects.filter(eleve=detail_eleve, matiere=matiere).first()
        matieres_notes.append({
            'matiere': matiere.nom,
            'note': note.valeur if note else 'NULL'
        })
```

```
    return render(request, 'notes/detail_eleve.html', {
        'eleve': detail_eleve,
        'matieres_notes': matieres_notes,
    })
```

Vue matiere()

```
def matiere(request, id):
    detail_matiere = get_object_or_404(Matiere, id=id)

    return render(request, 'notes/detail_matiere.html', {'matiere': detail_matiere})
```

Vue niveau()

```
def niveau(request, id):
    niveau_detail = get_object_or_404(Niveau, id=id)

    matieres = niveau_detail.matiere.all()

    return render(request,
    'notes/detail_niveau.html', {
        'niveau': niveau_detail, 'matieres':matieres
    })
```

Début des Templates

Fichier detail_eleve.html

```
    <h1>Détail de l'élève : {{ eleve.nom }}</h1>
<p>Niveau: {{ eleve.niveau.nom }}</p>
<p>Matières :</p>
<ul>
    {% for matiere in matieres_notes %}
    <li>{{ matiere.matiere }} - Note: {{ matiere.note }} </li>
    {% endfor %}
</ul>
```

Fichier detail_matiere.html

```
<h1>Détail de la matière</h1>
<p>Matière : {{ matiere.nom }}</p>
<p>Enseignant: {{ matiere.enseignant.nom }}</p>
```

Fichier detail_niveau.html

```
<h1>Détail du niveau </h1>
<p>Nom: {{ niveau.nom }}</p>
<p>Matières : </p>

<ul>
    {% for matiere in matieres %}
        <li>{{ matiere.nom }}</li>
    {% empty %}
        <li>Aucune matière associée à ce niveau.</li>
    {% endfor %}
</ul>
```

4. La ligne qu'on peut supprimer dans views(index).py est celui de **HttpResponse**

Navigations et espaces de noms

1.Reportons tout les templates comportant des détails, on a :

eleves.html

```
    <h1>Liste des élèves avec leurs notes et moyennes</h1><br>
<ul>
    {% for eleve in eleves %}
    <!-- <li>{{ eleve }}</li> -->
    <li>
        <strong>{{ eleve.nom }}</strong> - <a href="/notes/niveau/{{ eleve.niveau_id }}">Niv
        <ul>
            {% for matiere_note in eleve.matieres_notes %}
            <li>{{ matiere_note.matiere }}</li>
            {% endfor %}
        </ul>
        <li><a href="/notes/eleve/{{ eleve.id }}/">{{ eleve.nom }}</a></li><br><br><br>
    </li>
    {% endfor %}
</ul>
```

matieres.html

```
    <h1>Liste des matières</h1>
<ul>
    {% for matiere in matieres %}
    <a href="/notes/matiere/{{ matiere.matiere_id }}">
        <li>{{ matiere.matiere }}</li>
    </a>
    {% endfor %}
</ul>
```

2. On peut amélioré l'appel des vues en utilisant les espaces de nom que nous avions définit au niveau de notre fichier urls.py.

Fichier urls.java

```
from django.urls import path
from .views import index, eleve, matiere, niveau

app_name = "notes"

urlpatterns = [
    path('', index.index, name='index'),
    path('eleves/', eleve.eleves, name='eleves'),
    path('eleve/<int:id>/', eleve.eleve, name='eleve'),
    path('matieres/', matiere.matieres, name='matieres'),
    path('matiere/<int:id>', matiere.matiere, name='matiere'),
    path('niveau/<int:id>', niveau.niveau, name='niveau'),
]
```

On vas utilisé l'attribut **name** et notre **app_name** dans nos url qu'on vas définir ici , on a :

Templates eleves.html

```
    <h1>Liste des élèves avec leurs notes et moyennes</h1><br>
<ul>
    {% for eleve in eleves %}
    <!-- <li>{{ eleve }}</li> -->
    <li>
        <strong>{{ eleve.nom }}</strong> - <a href="{% url 'notes:niveau' eleve.niveau_id %}
        <ul>
            {% for matiere_note in eleve.matieres_notes %}
            <li>{{ matiere_note.matiere }}</li>
            {% endfor %}
        </ul>
        <li><a href="{% url 'notes:eleve' eleve.id %}">
           {{ eleve.nom }}
        </a></li><br><br><br>
    </li>
    {% endfor %}
</ul>
```

Templates matieres.html

```
    <h1>Liste des matières</h1>
<ul>
    {% for matiere in matieres %}
```

```
    <!-- <li>{{ matiere }}</li> -->
    <a href="{% url 'notes:matiere' matiere.matiere_id %}"><li>{{ matiere.matiere }}</li></a
    {% endfor %}
</ul>
```

3. La différence entre un projet et une applicaion est que le projet est l'ensemble de plusieurs applications tandis qu'une application est une partie d'un projet.

Le **projet** encapsule l'**application**. Parmi ces 2 termes, celui qui correspond plus à ce qu'on appel **notes** dans notre tp est application .

4. Pour créer des espaces de nom comme notes dans notre cas on vas faire comme suis, on a:

```
app_name = "notes"
```

Et comme sa on pourras définir autant d'application avec de différent nom d'espace.