TD N° 1 : Les opérations mathématiques avec Django

Ce TD a pour but de vous familiariser avec les concepts fondamentaux de Django, notamment la gestion des URLs dynamiques, la création de vues et l'affichage de résultats. En réalisant cet exercice, vous apprendrez à structurer un projet Django, à manipuler des données via des paramètres d'URL, et à fournir des résultats sous forme de réponse JSON ou de page HTML.

Vous devriez développer une petite application Django qui permet de réaliser des opérations mathématiques de base (addition, soustraction, multiplication et division) en utilisant une interface web. Les utilisateurs interagiront avec votre application via une URL structurée de la forme :

```
/operation/num1/num2/
```

Exemples :

```
/add/5/2.3    correspond à    5 + 2,3 = 7,3  
/sub/5/2.0    correspond à    5 - 2,0 = 3,0  
/mul/5.0/4.0  correspond à    5,0 * 4,0 = 20,0  
/div/15.0/3.0 correspond à    15,0 / 3,0 = 5,0
```

Où :

- `operation` est une chaîne de caractères indiquant l'opération à réaliser : `add`, `sub`, `mul` et `div`.
- `num1` et `num2` sont deux nombres réels sur lesquels effectuer l'opération.

Nous supposons que nous :

- Travaillons sous Windows 10 et utilisant PyCharm Community Edition.
- Avons créé un projet PyCharm ; c'est -à-dire, nous disposons d'un environnement virtuel créé par PyCharm et qu'il est actif et que nous sommes sous PowerShell dont voici l'invite de commande :

```
(.venv) PS D:\calculator\
```

Travail demandé : Donnez les commandes permettant de réaliser les opérations suivantes :

1. Créez un projet appelé `core` dans le répertoire du projet.
2. Créez une application nommée `calculator`.
3. Ajoutez l'application `calculator` au projet.
4. Configurez les URLs du projet pour inclure l'URLs de l'application.
5. Configurez les routes dynamiques dans `calculator/urls.py` pour ajouter les fonctionnalités souhaitées.
6. Programmez les calculs dans une vue au sein `calculator/views.py` selon que le résultat retourné est du JSON (API ou des interactions avec des clients front-end : AJAX, React, etc.) ou du HTML.
7. Pour le cas HTML, définissez le template `calculator.html` pour retourner le résultat et template `error.html` pour retourner les erreurs.
8. Tester l'application.

Remarques :

- A. Dans **Django 5**, lors de la définition des motifs avec **path()**, les convertisseurs disponibles sont:
 1. **str** : Capture une chaîne de caractères non vide, à l'exception du caractère /. C'est le type par défaut si aucun convertisseur n'est spécifié.
 2. **int** : Capture un nombre entier positif (y compris zéro) et le convertit en type entier.
 3. **slug** : Capture une chaîne composée de lettres ou chiffres ASCII, du trait d'union ou du caractère de soulignement. Par exemple, construire-votre-1er-site-django.
 4. **uuid** : Capture un identifiant UUID au format standard, incluant les tirets et en minuscules. Par exemple, 075194d3-6885-417e-a8a8-6c931e272f00. Il renvoie une instance de UUID.
 5. **path** : Capture toute chaîne non vide, y compris le séparateur de chemin /. Cela permet de correspondre à un chemin d'URL complet au lieu d'un segment de chemin d'URL comme avec **str**.

- B. Dans le présent projet utilisez **str** et faites la conversion dans la vue.

- C. Vous devriez gérer les erreurs au niveau de la vue.

- D. Exemples de gestions d'erreurs en python :

Erreur de valeur	<pre> try: chaine = 2.75 nombre = float(chaine) print(f"Conversion réussie : {nombre}") except ValueError: print("Erreur : la chaîne ne peut pas être convertie en float.") </pre>
Erreur quelconque	<pre> try: resultat = 10 / 0 except Exception as e: # Intercepte toute exception print(f"Une erreur s'est produite : {e}") </pre>