TP2 : Les Formulaires (ModelForm) dans Django

Objectif : Une fois que vos modèles sont créés et qu'un ensemble de leurs instances est affiché aux utilisateurs (clients) à l'aide des Templates. Il est temps écrire le code qui permet aux clients de réagir avec les informations affichées (rechercher une information, insérer une donnée…). Vous devrez donc mettre en place des formulaires avec des champs de saisie pour les utilisateurs de votre application.

Partie.1 : Création d'un formulaire à la main (sans ModelForm) :

Nous allons nous intéresser dans la première partie à la création des formulaires sans utiliser le ***ModelForm*** de Django. Vous allez utiliser des balises HTML pour insérer un formulaire de rechercher qui permet aux clients de rechercher l'existence d'un produit dans votre BDD. Les fichiers que vous allez modifier sont : ***urls.py*** - ***views.py*** et templates **/** ***search.html***

Travail demandé:

Afin d'afficher tous les produits correspondant à une recherche demandée par un client, vous allez :

1- Modifier le fichier ***search.html*** *(templates >> search.html)* pour insérer un formulaire permettant de la recherche.

```
<form action='' method="get">
  <label for="search"> Entrer le nom du produit: </label>
  <input type="text" name="recherche" id="search">
  <button type="submit"> demarrer la recherche </button>
</form>
```

2- Modifier le fichier **urls.py** (de l'application) pour définir l'***url*** sur laquelle vous allez répondre à la demande du client (afficher le résultat de la recherche) & la ***fonction*** (vue) qui va répondre à la demande (rechercher des produits).

```
urlpatterns = [  ...,
    path('list_produits/',views.rechercher_produits, name="listing"))
  ]
```

Il est clair que la fonction *rechercher_produits()* n'est pas encore définie, pour la définir il faut aller sur ***views.py***

3- Modifier le fichier ***views.py*** pour permettre d'interagir avec la BDD (***models.py)*** et de répondre à la requête par la suite, la réponse sera envoyé à un template (***search.html***) pour pouvoir l'afficher aux utilisateurs.

```
def rechercher_produits(request):
    produits=""
    if request.method =="GET":
        query=request.GET.get('recherche')
        if query:
            produits=Product.objects.filter(name__contains = query)

    return render(request,"search.html",{"product":produits})
```

4- Modifier le fichier ***search.html*** pour récupérer les données envoyées dans la fonction *rechercher_produits()* et l'insérer dans votre page html. Apres avoir l'insérer ces données seront accessibles par tous utilisateur à partir de l'url : http://127.0.0.1:8000/list_produits/

```
<h4> resultat de reherche : </h4>
<ul>
{% for produit in product %}
    <li>Nom du produit: {{produit.name}}, son prix: {{produit.price}} Da. </li>
  {% endfor %}
</ul>
```

Partie.2 : Création d'un formulaire à l'aide de ModelForm de Django :

Dans cette deuxième partie nous allons nous intéresser à la création des formulaires à l'aide de ***ModelForm*** de Django. Il s'agit d'un formulaire lié à un Model (Table), prenons par exemple un nouveau modèle : ***Commande***. Un client peut vous envoie une commande sur un produits, cette commande sera insérée dans votre base des commandes. Il est donc impératif de commencer par la création du *model*. Les fichiers que vous allez modifier sont:
models.py - ***forms.py*** - ***urls.py*** - ***views.py*** et templates / ***create.html***

Travail demandé:

Afin de permettre aux clients de nous envoie des commandes sur un produit, vous devrez :

1- Modifier le fichier ***models.py*** pour inclure la nouvelle classe *Commande* :

```
class Commande(models.Model):
      name=models.CharField(max_length=100)
      pproduit=models.ForeignKey(Product,on_delete=models.CASCADE)
```

2- Créer un nouveau fichier ***forms.py*** dans le même répertoire que votre application, modifier le pour définir votre classe qui va permettre de générer votre formulaire en la faisant dériver de *ModelForm* et en lui spécifiant le modèle à inclure *Commande*.

```
from django.db.models import fields
from django import forms
from .models import Commande

class CommandeForm (forms.ModelForm):
   class Meta:
      model = Commande
      fields="__all__"
```

3- Créer et modifier le fichier ***create.html*** *(templates >> create.html)* pour ajouter un formulaire permettant de l'insertion en précisant la méthode avec laquelle vous voulez envoyer les données insérées au serveur (POST ou GET).

```
<form action='' method="POST">

</form>
```

4- Modifier le fichier **urls.py** (de l'application) pour définir l'***url*** sur laquelle votre formulaire d'insertion sera accessible par les clients & la ***fonction*** (vue) qui va répondre à la demande (insérer une commande).
Avant de créer notre vue, ajoutons une configuration d'URL pour la page

```
urlpatterns = [  ...,
    path('create_cmd/',views.creer_commande))
  ]
```

Il est clair que la fonction *creer_commande()* n'est pas encore définie, pour la définir il faut aller sur ***views.py***

3- Modifier le fichier ***views.py*** pour décrire la vue (fonction) qui va traiter les données du formulaire, en principe c'est la même qui a servi à produire le formulaire, et transmettre par la suite l'instance ***form*** au contexte de gabarit (il s'agit du template ***create.html***).

```
def creer_commande(request):

    if request.method == 'POST':
        form = CommandeForm(request.POST)
        if form.is_valid():
            form.save()
            form = CommandeForm()
            mssg="commande envoyée, vous pouvez saisir une autre"
            # return redirect("listing")  # redirection vers la page de l'url: listing

        return render(request,"create.html",{"form":form,"message":mssg})

    else:
        form = CommandeForm()    #créer une instance de formulaire vierge
        mssg ="veuillez remplir tous les champs"

        return render(request,"create.html",{"form":form,"message ":mssg})
```

4- Modifier le fichier ***create.html*** pour récupérer le formulaire d'insertion *ModelForm* avec toutes ses fonctionnalités produites et envoyées dans la fonction *creer_commande()* et l'insérer dans votre page html. Apres avoir l'insérer ce formulaire sera accessibles par tous utilisateur à partir de l'url :
http://127.0.0.1:8000/create_cmd/

```
<h4> Formulaire pour passer des commandes : </h4>
<form  method="post">
    {% csrf_token %} <!--un jeton pour se protéger contre les attaques csrf -->
    {{ form.as_p }}
    <input type="submit" value="Valider">
</form>
```