# TP5 Python Django : Les formulaires

https://docs.djangoproject.com/en/5.1/

IFNTI-Sokodé L3

17 octobre 2024

Ce TP débute là où nous avions laissé le précédent. Nous allons créer des formulaires à partir des modèles et construire la page d'accueil du site. Ce TP est le plus important jusqu'à présent ! Il sera certainement nécessaire de prendre plus de temps que pour les précédents, mais ce n'est pas grave vu que c'est le plus important. Allez dans le bon répertoire et démarrez le serveur.

## 1 Mise en route, compréhension et création d'un formulaire

Les formulaires sont les parties les plus importantes, donc dangereuses, de tout site web. C'est depuis là que l'utilisateur peut modifier la base de données au travers de la méthode POST de HTTP. Nous allons créer des formulaires pour les modèles de notre application dont nous ne nous sommes pas occupés jusqu'à présent.

1. Qu'allons-nous devoir faire sur les vues de formulaire et sur le site en général pour éviter tout problème de base de données ? (Pensez au public ayant accès au site).
2. Avant de se concentrer sur les formulaires, ouvrez le site d'administration et visitez une page de modification d'un élève. Observez le champ `niveau` et les valeurs qu'il est possible de mettre dans `matieres`. N'y a-t-il pas un problème ? Lequel ? Nous verrons comment le régler dans la partie 3.
3. Dans tout site web issu de tout langage ou Framework, il est vraiment important de définir ce que peut voir et faire chaque type d'utilisateur sur le site et sur la base de données. Quels sont les différents types d'utilisateur qui peuvent accéder au site ? Donner au moins 3 types (mais 4 seraient parfaits) et expliquer ce que chacun d'entre eux peut voir et faire sur le site et sur la base de données. Pourquoi donc n'est-il pas nécessaire de construire des vues de formulaires sur les modèles que nous avons déjà créés ?
4. Quel modèle reste-t-il en termes de vues créées ? Quel(s) type(s) d'utilisateur pourront accéder aux vues de formulaires de ce modèle ? Que feront ces vues de formulaire ?
5. Créer un module views/note.py. Dans le fichier `note.py`, créer une vue pour ce modèle restant et nommez la `add_[nom_modèle]`. La fonction doit avoir 3 entrée. Lesquelles ? Pour le moment, dans cette fonction, écrivez simplement :
   "`return HttpResponse("Note d'un élève dans une matière.")`"
6. Dans le fichier `urls.py`, créer un chemin pour cette vue. Recopiez la ligne que vous avez ajoutée et expliquez à quel endroit du fichier elle se trouve. Vous pouvez tester si la page s'affiche.
7. Créer le `template` pour cette vue que vous nommerez `add_[nom_modèle].html`. Où l'avez-vous créé ?
   Faites en sorte que ce `template` permette à l'utilisateur de donner une note à un étudiant donné dans une matière donnée. Nous ne nous occupons pas de la base de données pour le moment, seulement du fichier HTML. Recopiez le contenu du fichier.
8. Dans le `template details_eleve`, pour chaque matière que l'élève suit, ajouter un lien vers cette vue avec les bons paramètres. Recopiez le contenu du `template`.
9. Changer la fonction dans le fichier `note.py` pour que :
   (a) Si la méthode de la requête est `POST`, un objet du modèle est créé en base de données.
   (b) Sinon une vérification que l'élève suit bien la matière est faite : si c'est le cas, l'affichage de la vue est "rendu", sinon, une `Exception` décrivant exactement le problème est levée.
   Recopier le contenu de la fonction et du `template`.
10. Pour tester la vue, il faut ajouter au moins une matière à l'élève. Ajoutez-la dans le site d'administration. Quelle matière avez-vous ajoutée à quel élève ? Testez la levée de l'`Exception` lorsqu'elle doit être levée et s'il est possible de voir la page quand cela doit être le cas.
11. Tester la vue pour voir si un objet est correctement créé lorsque cela doit être le cas. Insérez une capture d'écran de la page d'administration du modèle une fois qu'au moins une instance est créée.

## 2 Construction de formulaires de modèles

Dans la première partie, nous avons créé un formulaire à la main. Ce n'est pas une bonne façon de faire. Django peut s'occuper des formulaires pour vous, mais il faut lui dire quoi mettre dedans.

Créer d'abord un package nommé **forms** dans le dossier `notes`. Nous allons travailler sur les formulaires de modèles.

1. Saisie des notes
   (a) Dans ce package, créer un fichier `Noteform.py`, Ajouter ensuite dans ce fichier la classe **NoteForm** qui hérite de `ModelForm` (du package `django.forms`). Dans cette classe, créer une classe nommée `Meta` qui n'a ni argument ni parenthèse (i.e écrivez juste `class Meta:`). Dans celle-ci, créez trois attributs :
   — `model` qui prend `Note` comme valeur — `fields` qui prend `['valeur']` comme valeur
   — `labels` qui prend `{"valeur": "Note sur 20"}` comme valeur
   Recopier le contenu complet du fichier.
   (b) Changer la fonction associée à la vue de telle sorte que le formulaire soit automatiquement affiché (i.e. vous n'avez pas à préciser les champs de formulaire dans le `template`). N'oubliez pas de vérifier la validité du formulaire. Tester le bon fonctionnement (affichage et création d'un objet) et recopier le contenu de la fonction et du `template`.
   (c) Les valeurs d'une note sont comprises entre 0 et 20. Essayez de rentrer la valeur "-1" ou "21" dans le champ correspondant. Vous remarquerez que cela fonctionne. Pour régler ce problème, il faut ajouter des `Validators`. La meilleure pratique est de le faire dans le modèle, car il s'agit d'une contrainte de la base. 2 `Validators` sont à ajouter dans le modèle. Lesquels ? Ajoutez-les et recopiez le contenu du modèle.
   (d) Réessayer de rentrer "-1" ou "21" pour vérifier le bon fonctionnement et corrigez le cas échéant. Soyez attentif à bien comprendre d'où vient le problème ! Par exemple, il est possible que vous ne voyez pas d'erreur en validant le formulaire alors que la validation a été faite donc l'objet n'a pas été créé. Lorsque cela fonctionne, recopiez le contenu de la vue, du fichier `forms.py` et du `template`.
   (e) En fonction de la langue du navigateur, essayez de rentrer la valeur "10.5" ou "10,5" dans le champ de la note. Cela fonctionne ? Est-ce normal ? Ne modifiez rien, vu que cela dépend de la langue du navigateur.
   (f) **Pour aller plus loin** Cette question est un bonus. Nous avons presque terminé avec les formulaires. Maintenant que la page de création d'une note fonctionne, nous pouvons créer une page permettant d'ajouter des notes pour tous les élèves d'une même matière. C'est en fait ce qu'un enseignant voudrait faire, plutôt que de créer les notes une par une pour chaque élève. Faites-le dans une vue nommée `add_notes`. Il faudra ajouter la fonction, une ligne dans `urls.py` et un `template`.
2. Dans ce package, créer les modules EleveForm.py et EnseignantForm.py pour l'enregistrement d'un élève et d'un enseignant. Créer dans ces fichiers respectivement les formulaires **EleveForm** et **EnseignantForm**.
3. Définir les vues **add_eleve** et **add_enseignant** permettant respectivement l'inscription d'un élève et l'embauche d'un enseignant. Créer des urls associés à ces vues.
4. Définir les vues **update_eleve** et **update_enseignant** permettant respectivement la modification d'une instance Eleve et la modification d'une instance Enseignant. Créer des urls associés à ces vues.
5. Ajouter des liens sur chaque élève ou enseignant pour une modification.

# 3 La méthode clean

1. Qu'est-ce que la méthode `clean` ? Où doit-elle être déclarée ? Quelle est la première ligne à y inclure ?
2. Souvenez-vous de la question 1.2. Il y avait un problème sur les champs `niveau` et `matieres` d'un `eleve`. Il s'agissait d'une contrainte manquante qui peut être vérifiée à la création ou la modification d'un objet `eleve`. Lequel ? La méthode `clean` doit être déclarée sur un formulaire, car elle est invoquée lors de sa soumission.
3. Un élève ne peut pas être créé par un utilisateur lambda. Qui le peut ? Ce formulaire est donc uniquement destiné au site d'administration. Ouvrir donc le fichier `admin.py` et créer une classe nommée `EleveAdmin` héritant de `admin.ModelAdmin`. N'y ajouter qu'un seul attribut : `form = EleveForm`. Remplacer le `register` de `Eleve` par : `admin.site.register(Eleve, EleveAdmin)` Recopier le contenu du fichier.
4. Enfin, retournez au `EleveForm`, et créez une méthode `clean` vérifiant si les valeurs de `matieres` sont conformes au `niveau` donné dans le formulaire. N'oubliez pas que cette méthode est héritée ! Recopiez le contenu du fichier `forms.py`.
5. Ajouter un nouveau élève en tapant uniquement ou avec des caractères numériques. Que se passe-t-il ? Pourquoi ?
   En réalité les noms des personnes ne contiennent pas des chiffres. Alors créer une méthode `clean` pour prendre en compte. Faites-le pour enseignant aussi.

# 4 La méthode save

1. Qu'est-ce que la méthode `save` ? Où doit-elle être déclarée ? Quelle est la première ligne à y inclure ?
2. Créez une méthode `save` qui génère les matricules aux élèves et aux enseignants. Prendre :
   — Deux (02) premiers caractères du nom.
   — Deux (02) premiers caractères du prénom.
   — Le caractère du sexe
   — L'année de sa date de naissance.
3. Lorsqu'un étudiant est créé, il serait bien d'automatiquement associer les matières que cet étudiant suit grâce au niveau qui a été spécifié. Pour cela, nous devons autoriser des valeurs vides pour le champ `matieres` du modèle `Eleve`. Faites-le, définissez un `verbose_name` expliquant ce que l'utilisateur peut entrer et copiez la ligne que vous avez changée.
4. Créez une méthode `save` qui ajoute automatiquement les valeurs de `matieres` si le champ est laissé vide.

# 5 La page d'accueil

Le site est maintenant presque fonctionnel. Nous pouvons nous pencher sur la page d'accueil. Créez-en une (vue, `template`, etc.) facilement compréhensible par l'utilisateur. Recopier tout le code dans le rapport.