TP5 Python Django – Les vues

Librement tiré de la documentation officielle de Django version 3.1 :
`https://docs.djangoproject.com/fr/3.1/`

L3 - IFNTI Sokodé

10 Décembre 2020

Dans ce TP, nous allons nous focaliser sur la création de l'interface publique de notre site, grâce aux vues. À la fin de ce TP, nous aurons une vue pour consulter la liste des élèves avec leurs notes et leur moyenne dans chaque matière, une pour consulter le détail d'un élève, une pour consulter la liste des matières avec leur enseignant, une pour consulter le détail d'une matière et une pour consulter le détail d'un niveau.

1 Mise en route

Placez vous dans le bon dossier. Rappelez ce qu'est une vue. Sur quel fichier allons-nous principalement travailler lors de ce TP ? Vous pouvez lancer le serveur et ouvrir ce fichier.

2 Écriture de vues en code Python

1. Dans le fichier que vous venez d'ouvrir, vous devriez retrouver ce que l'on a fait lors du premier TP, qui écrit "Bonjour tout le monde !" sur la racine de l'application dans le navigateur. Il s'agira de la page d'accueil de l'application. Quelle est l'URL de la racine de l'application ?
2. Gardez pour l'instant la fonction telle quelle, et créez-en 5 autres correspondant aux 5 vues explicitées en introduction du TP. Nommez-les "`eleves`", "`eleve`", "`matieres`", "`matiere`" et "`niveau`" et faites les afficher un texte expliquant ce qu'elles produisent. Pour les 3 vues de détail, il faut rajouter un argument en entrée de la fonction. Quel est-il et à quoi sert-il ? Rajoutez cet argument et reportez l'ensemble du fichier.
3. Pour l'instant, ces vues ne sont pas accessibles. Que faut-il faire pour que ce soit le cas ? Dans quel fichier ? (Attention à ne pas confondre l'application et le projet, nous travaillons ici dans l'application `notes`). Ouvrez ce fichier, qui devrait déjà contenir le nécessaire pour accéder à la page d'accueil de l'application, et rajoutez-y ce qu'il faut pour pouvoir accéder aux différentes vues que vous avez créées (Doc : [chemin vers doc]/topics/http/urls). Les sous-URL seront les suivantes : `eleves/`, `eleve/[id]/`, `matieres/`, `matiere/[id]/` et `niveau/[id]/`. Reportez l'ensemble du fichier.
4. Rendez-vous aux différentes adresses qui sont maintenant accessibles. Comment faites-vous pour accéder aux pages de détail ?
5. Nous allons maintenant modifier les vues pour qu'elles accèdent aux données de la base de données. Pour chacune des pages, quel(s) modèle(s) devra(ont) être appelé(s) ?
6. Intéressons-nous d'abord à `eleves/` et `matieres/`. Dans les fonctions associées à ces 2 vues, faites en sorte d'afficher ce qu'elles doivent afficher de manière à peu près lisible (l'itération sur des ensembles de modèles fonctionne parfaitement). Le code pour interagir avec la base de données est le même que celui de l'API vu au TP précédent. Reportez le code des 2 fonctions.

3 Les templates

Vous ne trouvez pas cela bizarre d'écrire directement le contenu de la vue dans le code Python ?

1. Dans le répertoire `notes/`, créez le chemin `templates/notes/` et créez un fichier `index.html`. Ce fichier doit donc se trouver dans le dossier `ifnti_l3/notes/templates/notes/`. Qu'est-ce qu'un `template` ?
2. Dans ce fichier `index.html`, placez le code suivant : `<h1>Bonjour tout le monde !</h1>`. Que va faire ce code ?
3. Retournez dans le fichier `views.py` de l'application, et dans la fonction `index(request)`, remplacez le code par : `return render(request, "notes/index.html")`. Que fait `render` ? (Indice : il s'agit d'une "fonction raccourci") Regardez le résultat sur un navigateur.
4. Vous remarquerez que l'on n'utilise plus `HttpResponse` dans cette fonction. Modifier les vues `eleves` et `matieres` pour ne plus avoir à l'utiliser dans ces 2 vues non plus (n'oubliez pas les `templates`). Essayez de faire en sorte que les informations contenues dans les pages soient lisibles et que toutes les informations des modèles correspondants soient affichées (vous pouvez ne pas afficher les matières que les élèves suivent). Reportez le contenu du fichier `views.py` et des fichiers `templates` créés.

4 Gestion des erreurs

1. Passons maintenant aux autres vues. En quoi sont-elles différentes ?
2. Renseignez-vous sur la "fonction raccourci" `get_object_or_404`. Que fait-elle ? En quoi elle pourrait nous être utile pour les vues qu'il nous reste à écrire ?
3. Écrivez les vues restantes : le fichier `views.py` et les `templates`. Reportez le contenu de ces fichiers.
4. Une ligne peut maintenant être supprimée dans le `views.py`, laquelle ? Supprimez-la.

5 Navigation et espaces de noms

Il serait intéressant de pouvoir naviguer entre les pages du site. Par exemple, à chaque fois qu'un élément cité dans une page contient une page de détail, il faudrait pouvoir cliquer sur l'élément pour accéder à son détail.

1. Faites-le avec des balises `<a href="/notes/[chemin]/">` pour toutes les pages que vous avez déjà créées, et reportez tous les `templates`.
2. Comment pourriez-vous améliorer cela ? Faites-le pour tous les `templates` et recopiez le contenu de ces `templates` et du fichier `urls.py`.
3. Redonnez la différence entre un projet et une application. Lequel encapsule l'autre ? Parmi ces 2 termes, lequel correspond à ce que l'on appelle `notes` dans nos TP ?
4. En Django, il est possible de créer d'autres [réponse à la question précédente] (quelquefois même beaucoup d'autres !). Il est donc possible de retrouver le même nom pour une vue dans votre site et dans un autre. Pour éviter que cela pose problème, il faut créer des espaces de noms. Ceux-ci permettront à Django de savoir où chercher votre vue. C'est dans le fichier `notes/urls.py` que cela est à faire. Comment ? Faites-le avec le nom "`notes`" et ajoutez ce nom devant chaque URL appelées dans vos `templates` sous la forme `url 'notes:[nom_url]'`.

Ce TP est terminé :)