Exercices du 1er avril 2021

Flask – partie II

Ce soir nous allons apporter des améliorations à notre application Flask de la semaine passée. Premièrement, la liste des images va être transférée du fichier Json à une base de données relationnelle SQLite. Deuxièmement, l'ajout d'image va s'effectuer à l'aide d'une requête Ajax via la librairie JQuery. Troisièmement, nous allons ajouter de la validation (côté serveur et côté client).

1. Préparatifs et exploration (À faire avant le labo si ça n'a pas déjà été fait la semaine précédente)

- Assurez-vous d'avoir bien installé la librairie Flask;
- Téléchargez le code d'aujourd'hui dans un répertoire de travail. Tout se trouve, comme à l'habitude, sur le dépôt GitHub dans le dossier de la semaine;
- Testez l'application. Vous devriez être en mesure de démarrer le serveur Flask. Une visite sur le localhost avec votre navigateur vous retourne normalement une page web avec les formulaires.
- Explorez la structure de l'application : les dossiers et leur contenu, la liste des routes dans app.py, l'utilisation de Jinja dans les templates html, la structure du fichier Json de la liste des images, etc.
- Testez les 'routes supplémentaires' du fichier app.py. Utilisez votre navigateur pour interroger les routes et passer les paramètres dans la barre d'adresse. Modifiez au besoin les routes et les templates pour bien comprendre la mécanique!
- Regarder l'utilisation du moteur de template Jinja dans les fichiers base.html et allo.html. Comprenez-vous ce qui s'y produit? Voici un lien qui explique comment fonctionne l'héritage avec Jinja : https://jinja.palletsprojects.com/en/2.11.x/templates/#template-inheritance.
- Toujours concernant l'utilisation de Jinja, ouvrez le fichier listeImages.html pour avoir un aperçu du fonctionnement des structures itératives (boucle *for*) et conditionnelles (*if*).

2. Modifications à apporter à l'application

Modif 1 : Nous allons utiliser une base de données relationnelle SQLite au lieu d'un fichier json

- Au lancement de l'application, vous devez créer une base de données SQLite. Respectez la structure du fichier Json pour créer, en SQL, votre table *images*. Remplissez-la avec les données initiales du fichier json.
- Les routes 1 et 2 ne sont pas affectées par ce changement mais nous devons modifier les routes 3 et 4. Apportez les modifications nécessaires puis testez vos routes.
- La route 3 passe la liste entière des images au template listeImages.html et laisse Jinja faire le travail de filtrer selon la présence ou l'absence de droits d'auteurs. Nous allons plutôt filtrer la liste avant de la passer au template, puis modifier le script Jinja en conséquence (enlever la structure conditionnelle).

Modif 2 : Nous allons utiliser une requête Ajax pour l'ajout d'image

- En javascript, dans le template formulaire.html, écrivez une fonction qui exécute une requête Ajax à l'aide de la librairie JQuery. Cette requête doit permettre l'ajout d'une image en utilisant la route appropriée dans app.py. N'oubliez pas d'importer la librairie par son CDN dans le template base.html.
- Liez ensuite l'évènement click du bouton *Ajouter* à votre fonction à l'aide d'un *Event listener*. Pour empêcher l'envoie par défaut du formulaire en mode synchrone, utilisez la méthode preventDefault(); renseignez-vous en ligne sur son utilisation.
- Dans app.py, modifiez le retour de ajoutImage() pour l'envoie d'un message confirmant l'ajout de l'image à la base de données. Votre requête ajax doit recevoir ce message et le transmettre à l'utilisateur (via la fonction alert() par exemple).

Modif 3 : Validation côté serveur

- Toujours dans la fonction ajoutImage(), nous allons valider la présences des 3 éléments (fichier, titre et droits) avant d'ajouter l'image à la base de données et d'enregistrer l'image sur le serveur.
- Le message retourné peut aussi être ajusté en fonction du résultat de la validation. Retournez un message de succès ou d'échec selon le cas.

Modif 4 : Bonus! Validation côté client

- Cette partie est à faire de votre côté. Plusieurs options s'offrent à vous pour valider un formulaire côté client :
    a. Vous pouvez utiliser les fonctionnalités de Bootstrap qui permettent d'obtenir un résultat très professionnel. Voir : https://getbootstrap.com/docs/5.0/forms/validation/
    b. Vous pouvez écrire une fonction en javascript qui s'exécute avant votre requête ajax et qui valide les champs du formulaire.
    c. Parfois la simple utilisation de l'attribut 'required' dans un élément html suffit pour valider la présence d'une valeur dans un champs.