Exercices du 25 mars 2021

Flask

La semaine dernière nous avons créé une page web statique en n'utilisant que du html, du css et du javascript (avec les librairies Bootstrap et JQuery). Cette page interrogeait un serveur distant à l'aide de requêtes Ajax et nous utilisions du javascript pour générer du html en fonction des données retournées. Aussi, pour lier l'évènement *click* d'un bouton à l'exécution de notre requête, nous avions créé un *Event Listener* en javascript.

Cette semaine, nous allons créer un site web dynamique possédant son propre Backend codé en Python à l'aide de la librairie Flask et du moteur de template Jinja. La page principale permet soit d'ajouter une nouvelle image dans la banque d'images, soit d'interroger la banque d'images pour en obtenir la liste. Nous utiliserons encore Bootstrap pour le formatage visuel de l'interface utilisateur mais nous n'aurons pas besoin de JQuery. Notez qu'au lieu de lier l'évènement *clic* des boutons à un *Event Listener* en javascript, nous utiliseront les attributs de l'élément *form* pour générer une requête vers le backend. Cette requête n'étant pas asynchrone (i.e. pas du ajax), la page sera rechargée à chaque appel. Finalement, pour stocker de façon persistante l'information concernant les images, nous utiliseront un fichier au format *Json* mais nous aurions aussi pu utiliser une base de données relationnelle comme par exemple *SQLite*.

1. Préparatifs

- Assurez-vous d'avoir bien installé la librairie Flask;
- Téléchargé 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 '…'.

2. Exploration

- 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*).

3. Programmation des routes

Route 1 : '/' selon la méthode GET

- Vous devez retourner le template formulaire.html. Consultez la documentation Flask et prenez en exemple le code des routes supplémentaires.
- En allant à la racine du localhost, vous devriez voir apparaitre la page Formulaires dans votre navigateur. Les boutons ne sont pas encore fonctionnels, nous allons y remédier au cours de l'exercice.

Route 2 : '/img/<path:path>' selon la méthode GET

- Vous devez retourner le fichier correspondant (l'image) s'il existe. Consultez la documentation Flask et les exemples de code donnés par Alix.
- En inscrivant l'adresse d'une image existante dans la barre d'adresse du navigateur, vous devriez voir apparaître cette image (ex. localhost :5000/img/escargot.jpg).

Route 3 : '/listeImages' selon la méthode GET

- Cette route demande de faire un peu de traitement avant de retourner le template listeImages.html. Vous devez entre-autres lire le fichier Json et stocker son contenu sous forme de liste de dictionnaires. Au moment de retourner le template, passez en argument nommé cette liste ainsi que la valeur de l'argument *droits*. Assurez-vous que cette dernière a été convertie en booléen; le plus facile est d'utiliser une expression conditionnelle!
- Vous devriez maintenant être en mesure, à partir de la page principale, de faire apparaître la liste des images selon l'option de droits d'auteur voulu, puis de faire apparaître une image en cliquant sur un des liens dans la liste. Prenez le temps de débugger votre code au besoin. Faites-moi signe dans le *chat* quand vous avez terminé cette étape ou qu'il ne vous reste plus qu'à débugger!!

Route 4 : '/ajoutImage selon la méthode POST

- Cette route consiste à enregistrer le fichier de l'image dans le dossier static/img, ajouter une référence à cette image dans le fichier Json, puis retourner la page principale. Ici aussi la documentation Flask et une recherche sur StackOverflow vous seront utiles. Pour l'ajout d'un élément au fichier Json, vous pouvez stocker son contenu dans une liste de dictionnaires, ajouter un élément à la liste, reconvertir en Json puis remplacer le contenu du fichier.
- Testez la route en utilisant le formulaire pour ajouter des images puis vérifiez les modifications apportées au fichier Json et au contenu du dossier img. Faites apparaître la liste des images pour confirmer le bon fonctionnement.
- Comme exercice supplémentaire, vous pourriez implémenter une validation pour forcer l'utilisateur à bien remplir les champs avant de soumettre une nouvelle image!