Activité : application Todo liste avec un serveur et une seule page HTML

Présentation

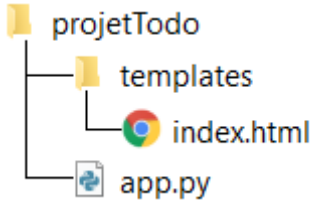
Cette première activité montre comment créer une **application Todo liste** qui affiche **une seule page html** avec un **contenu statique**.

Préparation du projet

- je crée un dossier **projet_** ;
- dans le dossier **projetTodo** je crée le fichier python de l'application **app.py** ;
- dans le dossier **projetTodo**, je crée un sous-dossier **templates** ;
- dans le sous-dossier **templates** je crée une page **index.html**

> Le sous-dossier **templates** va contenir toutes les pages html du projet

Le résultat à obtenir



Le fichier python de l'application

- Ecrivez le contenu suivant dans le fichier **app.py** : <file python app.py> #! /usr/bin/python # -- *coding:utf-8* --

from flask import Flask, render*template, request app = Flask(name_*)

@app.route('/') def index():

```
  return render_template('index.html')
```

if name == 'main':

```
  app.run(debug=True)
```

</file>

Explications des instructions

- l'instruction from permet d'importer
  - le module **Flask** qui gère le serveur HTTP ;
  - le module **render_template** qui permet d'utiliser les pages HTML comme modèle ou **template** ;
  - le module **request** pour gérer les **envois** de données avec la **méthode GET ou POST** ;
  - la création de la variable **app** est fondamental car il s'agit de l'application Web ;
  - **app.run(debug=True)** permet de **lancer** l'application et donc le serveur Web. Le paramètre **debug** lance l'application dans ce mode car cela est **utile lors du développement** de mon application pour **détecter les erreurs** et permettre de **rafraichir** les pages web.
  - la fonction **index** correspond à une **vue**. Elle peut porter le nom que je veux :
    - elle permet au serveur de **répondre** à une requête à l'adresse, **la route**, indiquée à la ligne précédente
    - cette ligne située juste avant la fonction **index** est un décorateur (**@app.route**) qui indique :
      - quelle **route** (adresse) permet d'exécuter la fonction **index** et donc au serveur de **renvoyer** sa **réponse** ;
      - de **préparer la bonne exécution** de la fonction **index**

- la fonction **index** va utiliser le **moteur de template** (modèle) de Flask qui s'appelle **Jinja2**, pour utiliser la page passée en paramètre comme modèle de réponse et, comme je verrai plus tard, permettre de personnaliser la page avant son envoi au navigateur.

Une fonction **décorée** par **@app.route** pour renvoyer une page Web est appelée **vue**.

ATTENTION :
Mon application **ne doit pas avoir** deux routes identiques, ou deux vues portant le même nom !

Le fichier HTML de la page Web

- Ecrivez le contenu suivant dans le fichier **index.html** : <file html index.html> <!DOCTYPE html> <html lang="fr"

>

<

head> <meta charset="UTF-8"

>

```
<title>Application Todo</title>
```

</head>

<

body> <h1>Todo liste</h1>

Ajouter une nouvelle tâche :

```
  <form action="/ajout" method="POST">
    <input type="text" name="tache">
    <input type="submit"></div>
  </form>
</div>
<div>
  <ul>
    <li><input type="checkbox">Tâche 1</li>
    <li><input type="checkbox">Tâche 2</li>
    <li><input type="checkbox">Tâche 3</li>
  </ul>
</div>
```

</body> </html> </file>

Explications des instructions

- la page HTML contient la balise **<form>** qui définit un **formulaire** pour permettre l'**envoi d'informations** au serveur. Le formulaire contient :
  - un **attribut action="/ajout"** qui précise la **route** recherchée sur le serveur ;
  - un attribut **method="POST"** qui précise la **méthode** d'envoi des données ;
  - un champ de saisie d'information **<input type="text">** qui a pour nom **tache** ;
  - un bouton **<input type="submit">** pour soumettre (envoyer) le formulaire au serveur.

Lancement de l'application

- j'ouvre une invite de commandes puis je lance l'exécution de l'application **app.py**. Je dois avoir le résultat suivant :

```
D:\...\projet_1>python app.py
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: Do not use the development server in a production environment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 881-708-049
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

En cas d'erreur de ce type :

```
    self.socket.bind(self.server_address)
OSError: [WinError 10013] Une tentative d’accès à un socket de manière interdite par
ses autorisations d’accès a été tentée
```

- je **modifie** le fichier de l'application **app.py** pour utiliser un **port réseau différent** (par exemple 5005) du port par défaut (5000) en modifiant la ligne suivante :

```
if __name__ == '__main__':
    app.run(debug=True, port=5005)
```

Dès que j'ai lancé l'application, je **ne peux plus saisir de commandes** dans l'invites de commandes car l'application est lancée. Je ne **dois pas fermer la fenêtre** sinon l'application de fonctionne plus.

```
* je lance mon navigateur, et je me rends à l'adresse http://127.0.0.1:5000. Je dois avoir le
résultat suivant :
```

- le **modifie** le contenu de ma page **index.hmtl** puis le **réactualise** la fenêtre du navigateur pour **visualiser** les **modifications**.

Retour à l'activité : Utiliser le framework Flask pour créer un site Web dynamique

- Activité : Utiliser le framework Flask pour créer un site Web dynamique