Cours NSI: Client-Serveur

1 Introduction

Afin de travailler sur les interactions entre client et serveur sur le web, nous allons créer notre propre serveur web grâce à python et au module/bibliothèque Flask.

2 Compétences

- Distinguer ce qui est exécuté sur le client ou sur le serveur et dans quel ordre.
- Distinguer ce qui est mémorisé dans le client et retransmis au serveur.
- Analyser le fonctionnement d'un formulaire simple.
- Distinguer les transmissions de paramètres par les requêtes POST ou GET.

3 Mise en place du serveur Flask

Commencez par lancer le logiciel "Thonny".
Rendez vous maintenant dans le menu "Outils" puis cliquez sur "Gérer les paquets".
Dans la barre de texte en haut, tapez "flask" une fois le paquet trouvé, installez le.
Maintenant, créez un dossier et enregistrez un fichier python contenant le code ci-dessous dedans puis lancez-le.

```
from flask import Flask

app = Flask(__name__) # Création d'un objet Flask, necessaire au lancement du
    serveur.

@app.route('/') #Permet d'indiquer que la fonction suivante est liée à une
    requête sur le serveur à l'adresse racine.
def index(): #Un nom de fonction comme un autre
  return "<p>Bonjour le monde !</p>" #On renvoie la page.(ou le contenu du body
    si ce n'est pas une page entière).

app.run(debug=True)
```

Si tout c'est bien passé jusqu'ici vous devriez voir le message suivant dans la console: "* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)".
Ci c'est le cas, allez sur le lien et vérifiez que le site affiche bien le message.

3.1 Pratique

Créer une autre page avec une route différente, puis s'y connecter en allant sur la page. Pour y accéder il faut modifier l'adresse pour avoir le même contenu que celui précisé dans la partie @app.route(...) que vous aurez défini.

4 Des pages un peu plus complexes.

Écrire une page web entière dans un return ne semble pas très pratique, et en effet nous allons voir comment renvoyer une page web préparée dans un fichier HTML.
Commencez par créer un dossier "templates" dans le dossier ou votre fichier python est actuellement. Dans ce dossier, créer un fichier texte et nommez le "index.html" puis mettez le contenu suivant (ou un autre fichier HTML que vous avez déjà créé) dans le fichier.

```
<!doctype html>
<html lang="fr">
  <head>
    <meta charset="utf-8">
    <title>Page d'accueil</title>
  </head>
  <body>
    <h1>Ceci est un site</h1>
    <p>Voici mon site, il fonctionne bien mais manque un peu de contenu.</p>
  </body>
</html>
```

Maintenant, il n'y a plus qu'à changer le return de la fonction index par ceci:
`return render_template("index.html")`
Et changer l'import du début : `from flask import Flask, render_template`

5 Exécution coté serveur.

Ajoutons quelques éléments pour que notre serveur puisse changer une page.
au début du fichier :
`from random import randint`
Dans le code HTML : `<p>Nombre gagnant:{{nb}}</p>`
dans le render_template:
`return render_template("index.html", nb=randint(0,100))`
On peut observer que le choix du nombre est bien fait du coté serveur en inspectant la page html, il n'y a que le nombre.
Flask fait le lien entre l'argument nommé nb dans le render_template et le mot entre {{ et }} dans le HTML.

6 POST, GET, ...

Les requêtes POST et GET sont deux types de requêtes qu'un client peut faire lors de l'accès à un serveur. De manière générale les requêtes POST servent à envoyer des données et GET à en recevoir. Néanmoins, les deux peuvent être utilisées pour envoyer des données au serveur (par exemple avec un formulaire).

Lors de l'envoi des données, les deux se comportent différemment :

- Les données envoyées avec une requête GET se retrouvent dans l'adresse du site web par exemple, lorsqu'on fait une recherche avec Google. https://www.google.com/search?**q=marecherche**
- Les données d'une requête POST ne sont pas transmise avec l'adresse, mais directement au serveur. Il s'agit d'une méthode plus sécurisée.

Nous allons voir le fonctionnement d'un formulaire avec une requête POST: Commençons par rajouter un élément à notre import :
`from flask import Flask,render_template,request`
Maintenant, ajoutons le code pour gérer les nouvelles pages de notre site :

```
@app.route('/formulaire')
def form():
    return render_template('formulaire.html')

@app.route('/donnees/', methods = ['POST', 'GET'])
def data():
    if request.method == 'GET':
        return f"Veuillez passez par /formulaire"
    if request.method == 'POST':
        form_data = request.form
        # on peut acceder directement aux données envoyées dans le formulaire
     avec form_data["nom_de_la donnée"] par exemple form_data["Ville"]
        return render_template('donnees.html',form_data = form_data)
```

Il ne reste plus qu'à créer les deux fichiers HTML :
formulaire.html

```
<!doctype html>
<html lang="fr">
  <head>
    <meta charset="utf-8">
    <title>Page daccueil</title>
  </head>
  <body>
    <form action="/data" method = "POST">
      <p>Nom <input type = "text" name = "Nom" /></p>
      <p>Ville <input type = "text" name = "Ville" /></p>
      <p>Pays <input type = "text" name = "Pays" /></p>
      <p><input type = "submit" value = "Submit" /></p>
    </form>
  </body>
</html>
```

donnees.html

```
<!doctype html>
<html lang="fr">
  <head>
    <meta charset="utf-8">
    <title>Page daccueil</title>
  </head>
  <body>
    {% for key,value in form_data.items() %}
    <h2> {{key}}</h2>
    <p> {{value}}</p>
    {% endfor %}
  </body>
</html>
```