Introduction aux services Web

Alexis GUILLOTEAU

IMT Lille Douai
École Mines-Télécom
IMT-Université de Lille

TP - Introduction aux services Web et bases de données

Introduction de Flask

Flask est un framework open-source de développement web en Python. Son but principal est d'être léger, afin de garder la souplesse de la programmation Python, associé à un système de templates. Flask est flexible et compatible avec la majorité des systèmes de base de données. Son fonctionnement repose sur une architecture REST. Un équivalent de Flask peut être Django.

Configuration de l'environnement et exécution du webservice

Le web service devra être exécuté avec les droits administrateur sur votre machine, cela peut être par le lancement de votre IDE en mode administrateur ou directement par votre terminal tel qu'une commande sudo sous linux.

En étant placer dans votre dossier de projet vous pourrez ajouter les variables d'environnement avec les commandes suivantes :

```
set FLASK_APP = app.py
set FLASK_ENV = development
```

Son démarrage se fera avec la commande :

```
flask run
```

Exécution d'appel sur votre web service

Pour tester votre web service, deux méthodes s'offre à vous :

- Utiliser un navigateur web et atteindre les routes du web services
- Utiliser des commandes « curl »

Curl peut nécessiter une installation sur votre machine.

Import nécessaire

```
from flask import Flask
from flask import request
from flask import render_template
from flask import flash, redirect, url_for, send_file
import os
```

Attention, ces imports sont ceux de base, lors de ce TP des erreurs seront présentes à cause de manque d'import, cela est intentionnel. Vous devrez rechercher les imports manquants.

Ajout d'une route

Le fonctionnement de Flask se fait par l'intermédiaire de « routes ».
Ces routes correspondent à des URL de votre application. Le chemin de cette URL, la méthode utilisée (GET, POST…) et les paramètres à entrer seront configurables.

Déclaration de votre application

Flask défini vos web services en temps qu'application.

Rappel : Python est sensible à l'indentation que vous choisissez. Je vous conseille un système par 4 espaces ou par tabulation. Gardez la méthode choisie dans l'intégralité de votre fichier.

Déclarer l'application :

```
app = Flask(__name__)
```

Créer votre première route associée à cette application :

```
@app.route('/bonjour')
def bonjour():
    return
```

La route décrite ci-dessous sera atteinte par l'adresse « ip/bonjour ». Elle démarrera la routine « bonjour ». Il n'y a pas de retour spécifique du web serveur, la méthode d'appel n'est pas testé et aucun paramètre n'est nécessaire.

Une fois la route créer, modifier la afin de retourner une chaîne de caractères :

```
@app.route('/bonjour')
def bonjour():
    return 'hello world'
```

Démarrer un navigateur web et essayer l'adresse suivante :

http://127.0.0.1:5000/bonjour

Vous devriez avoir comme retour la phrase « hello world ».

La commande curl correspondante sera :

curl 127.0.0.1:5000/bonjour

Requête GET

Nous allons maintenant passer un argument sous la forme d'une requête GET.

Ajoutez une nouvelle route telle que :

```
@app.route('/getit')
def getit():
    argument = request.args.get('argument')
    argument = argument + " OK"
    return argument
```

Si par navigateur vous utilisez l'URL :

http://127.0.0.1:5000/getit?argument=test

La réponse sera :

Test OK

Par curl la requête est :

curl 127.0.0.1:5000/getit?argument=test

Un second argument peut être rajouter en completant la requête telle que :

curl 127.0.0.1:5000/test?arga=test&argb=test

Programmer une route permettant l'addition de deux paramètres et le résultat comme réponse.

Programmer une route permettant la comparaison de deux chaînes de caractères (longueur, caractères similaires).

Réception d'un fichier et requête POST

A la suite de votre projet, ajouter la méthode suivante :

```
@app.route('/postexample',methods=['POST'])
def postexample():
    name_file = "test.png"
    if request.method == 'POST':
        file = request.files['file']
        file.save(UPLOAD_FOLDER+name_file)
        return send_file("result.png",mimetype='image/png')

    return "NOK"
```

Cette méthode teste si la requête est de type POST, récupère un fichier, l'enregistre et vous en renvoi une autre.

L'enregistrement de ce fichier se fera par l'intermédiaire d'un chemin préenregistrer « UPLOAD_FOLDER ».

Cette route sera testée par curl de la façon suivante :

curl -F file=@D:\UV\PycharmProjects\...\venv\TEST\input.png 127.0.0.1:5000/postexample --output result.png

Attention : Si vous êtes sur Windows, vous devrez ouvrir votre invité de commande dans un répertoire safe comme « Mes documents » si vous voulez avoir les droits de récupérer l'image « result.png ».

La prochaine méthode permettra le traitement différencié selon le type de contenu (texte pur ou json).

```
@app.route('/postarg',methods=['POST'])
def postarg():
    if request.headers['Content-Type'] == 'text/plain':
        return "OBWK: " + str(request.data)
    elif request.headers['Content-Type'] == 'application/json':
        return "key1: " + str(request.json["key1"])

    return "NOK"
```

Essayer les deux requêtes suivantes et expliquer le fonctionnement :

curl -d "Hello there" -H "Content-Type: text/plain" -X POST http://127.0.0.1:5000/postarg

curl -d "{\"key1\":\"value1\", \"key2\":\"value2\"}" -H "Content-Type: application/json" -X POST http://127.0.0.1:5000/postarg

Modifier la route de façon à accepter des informations de login et mot de passe dans un json, pour l'instant toujours répondre « Compte en attente de validation » si un json avec des informations de login ET de mot de passe non vides sont présentes.

On peut très bien imaginer envoyer un (ou des) fichier à un serveur avec un ID bien spécifique puis grâce à une requête suivante demander le traitement.

Gestion de base de données

Créer une nouvelle route qui permettra la création d'une base de données :

```
@app.route('/createmanu')
def createmanu():
    conn = sqlite3.connect('database.db')
    print ("Opened database successfully")
    #type : https://www.sqlite.org/datatype3.html
    conn.execute('CREATE TABLE students (name TEXT, addr TEXT, city TEXT, pin
TEXT)')
    print("Table created successfully")
    conn.close()
    return 'Your table is created.'table is created.'
```

Cette route sert à créer une base contenant uen table de donnée « etudiants » contenant les informations suivantes sous 4 champs texte :
Nom ; adresse ; ville ; code personnel

En utilisant « DB Browser for SQLite » vous pouvez ouvrir le fichier « database.db » afin de visualiser et modifier votre base manuellement.

Suite à la création de cette base nous allons ajouter une route qui permettra à un étudiant de s'ajouter.

```
@app.route('/addmanu')
def addmanu():
    conn = sqlite3.connect('database.db')
    cursor = conn.cursor()
    oneRecord = ['Sophie','Boulevard Concorde','Paris','Xae-A12']
    cursor.execute('INSERT INTO students VALUES (?,?,?,?)', oneRecord)
    conn.commit()
    conn.close()
    return 'OK'
```

Tester cette méthode puis modifier là afin de prendre en compte une requête POST avec json.

En suivant la documentation python de flask et sqlite, ajouter des routes permettant la suppression d'un étudiant et la récupération des informations d'un étudiant. Si l'étudiant n'existe pas on doit avoir un message d'erreur. Modifier aussi votre première route d'ajout pour ne pas autoriser l'insertion d'un étudiant déjà existant.
Enfin mettre en place une procédure de mise à jour d'un étudiant.

L'étudiant doit pouvoir acceder à une url d'inscription dont un des paramètres sera le mot de passe.

Créer une seconde table avec comme clef primaire un ID aléatoire unique pour chaque étudiant (à relier à l'étudiant) contenant une série de notes pour l'étudiant.

L'étudiant doit pouvoir récupérer des (toutes ou certaines) par l'intermédiaire d'une requête avec JSON et mot de passe. Le serveur doit répondre des messages d'erreurs selon les situations (étudiant inconnu, mot de passe incorrect…) et réenvoyer une JSON comportant les notes demandées.

Finalement l'étudiant doit pouvoir mettre à jour sa photo et la récupérer.

Indice : INSERT, SELECT, DELETE, UPDATE