10.7 Créer un serveur avec flask

NSI 1ère - JB Duthoit

Nous allons ici créer ici un serveur web installé sur votre ordinateur.
☛ Ici, le client et le serveur vont se trouver sur la même machine !
Nous aurons donc 2 logiciels sur le même ordinateur : le client (navigateur web) et le serveur (serveur web), ces 2 logiciels vont communiquer en utilisant le protocole HTTP.
Nous allons pour cela travailler avec le framework Python Flask.
Ce framework va nous permettre de générer des pages web côté serveur, il possède son propre serveur web.

10.7.1 Exemple de base

Nous allons commencer par un cas très simple où le serveur va renvoyer au client une simple page HTML statique.

Créez un fichier Python `"essai_flask.py"` (ce fichier devra être sauvegardé dans un répertoire nommé "flask" précédemment créé). Saisissez le code suivant dans le fichier `"essai_flask.py"` :

```
from flask import Flask  #import bibliothèque flask
app = Flask(__name__) #Création d'un objet app

@app.route('/')  #index sera réalisé si le serveur web recevra
   une requête HTTP avec une URL correspondant à la racine du
   site ('/')
def index():
  return "<p>Hello World !</p>"
app.run(debug=True) # Pour lancer le serveur
```

Après avoir exécuté le programme ci-dessus, ouvrez votre navigateur web et tapez dans la barre d'adresse `"localhost:5000"`. Qu'est ce qui s'affiche ?

Remarque

☛ Avec le "localhost", on indique au navigateur que le serveur web se trouve sur le même ordinateur que lui (on parle de machine locale)
☛ En ouvrant un navigateur web et en tapant `"localhost:5000"`, nous faisons une requête HTTP, le serveur web fourni avec Flask répond à cette requête HTTP en envoyant une page web contenant uniquement `"<p>Hello world !</p>"`

Modifier le fichier comme ceci :

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def index():
  return "<p>Hello World !</p>"
```

```
@app.route('/page')
def about():
  return "<p>Hello !!!</p>"

app.run(debug=True)
```

☛ Ouvrer votre navigateur, et saisir "`localhost:5000`", puis "`localhost:5000/page`"

10.7.2 En utilisant un template

Dans votre répertoire "Flask", créez un répertoire "`templates`".
Dans ce répertoire `templates`, créez un fichier `index.html`.

Saisir quelques lignes html afin d'avoir une page web de quelques lignes. Pour rappel, la structure :

```
<!doctype html>
<html lang="fr">
  <head>
    <meta charset="utf-8">
    <title>Mon titre</title>
  </head>
  <body>
    <h1>Mon site</h1>
    <p>bla bla ...</p>
  </body>
</html>
```

Modifier le programme `essai_flask.py` comme ceci :

```
from flask import Flask,render_template # importation de
   render_template
app = Flask(__name__)

@app.route('/')
def index():
    return render_template("index.html")

app.run(debug=True)
```

☛ Le serveur renvoie maintenant au client la page HTML correspondant au fichier "`index.html`" qui a été créé dans le répertoire "`templates`".

⚠ Attention, il est obligatoire d'utiliser un répertoire `templates`.

Remarque

Pour ajouter du style `css` à votre page html :

- Créer dans votre dossier `flask` un sous-dossier `static`
- Créer dans votre dossier `static` un sous-dossier `styles`
- Mettre le fichier `css` dans votre dossier `styles`.
- Modifier votre fichier `html` en conséquence...

10.7.3 Pages dynamiques

Créer le fichier "`index.html`" de cette façon :

```
<!doctype html>
<html lang="fr">
  <head>
    <meta charset="utf-8">
    <title>Mon formulaire</title>
  </head>
  <body>
    <form action="http://localhost:5000/result" method="post">
        <label>Nom</label> : <input type="text" name="nom" />
        <label>Prénom</label> : <input type="text" name="prenom
 " />
        <input type="submit" value="Envoyer" />
    </form>
  </body>
</html>
```

et créer un fichier "`resultat.html`" :

```
<!doctype html>
<html lang="fr">
  <head>
    <meta charset="utf-8">
    <title>Result</title>
  </head>
  <body>
    <p>Hello {{prenom}} {{nom}} !</p>
  </body>
</html>
```

Modifiez le fichier `essai_flask.py` comme ceci :

```
from flask import Flask, render_template, request
app = Flask(__name__)

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

```
@app.route('/result',methods = ['POST'])
def resultat():
  res = request.form
  n = res['nom']
  p = res['prenom']
  return render_template("resultat.html", nom=n, prenom=p)

app.run(debug=True)
```

Remarque

On remarque 2 attributs dans cette balise form :

- `action="http://localhost:5000/resultat"`
- `method="post"`.

Ces 2 attributs indiquent que le client devra effectuer une requête de type POST dès que l'utilisateur appuiera sur le bouton "Envoyer".
Cette requête POST sera envoyée à l'URL "`http://localhost:5000/resultat`".

☛ "`request.form`" est un dictionnaire Python qui a pour clés les attributs "`name`" des balises "input" du formulaire (dans notre cas les clés sont donc "nom" et "prenom") et comme valeurs ce qui a été saisi par l'utilisateur.

Exercice 10.159
Modifier les fichiers précédents afin d'utiliser la méthode `get` au lieu de la méthode `post`. Observer la différence au niveau de la barre d'adresse.
⚠ Attention, avec la méthode `GET`, il faut utiliser `request.args` au lieu de `request.form` !

Exercice 10.160
Utiliser `Flask` pour gérer votre formulaire créé dans la partie précédente. Pour rappel, votre questionnaire contient quelques questions au sujet de l'institut de Genech.
Ajouter à ce formulaire un champ pour récupérer votre nom et votre prénom.
Ajouter à ce formulaire un bouton 'Envoyer", et créer un serveur local avec le framwork Flask qui permet d'avoir un retour du questionnaire. Il faudra pour cela utiliser un template.
Le type de retour est un message lié au pourcentage de bonne réponse. Pour exemple :

- Si l'utilisateur n'a fait aucune erreur, le message doit être du type "Bravo nom prénom, vous avez 100 % de réussite"".
- Si l'utilisateur n'a fait qu'une erreur, le message doit être du type "Pas mal nom prénom, vous n'avez qu'une seule erreur" ...
- et ainsi de suite...

Exercice 10.161
On ajoutera à la page renvoyée par le serveur, en plus du score globale, la correction du qcm...

Exercice 10.162
On modifiera la page renvoyée par le serveur. Pour chaque question :

- Si l'utilisateur répond correctement, la réponse sera notée en vert
- Si l'utilisateur se trompe, sa réponse sera notée en rouge, et la bonne réponse en vert.