Partie 1

Flask : site Web côté serveur

Flask est une *bibliothèque* python fournissant les outils pour faire fonctionner un serveur web.

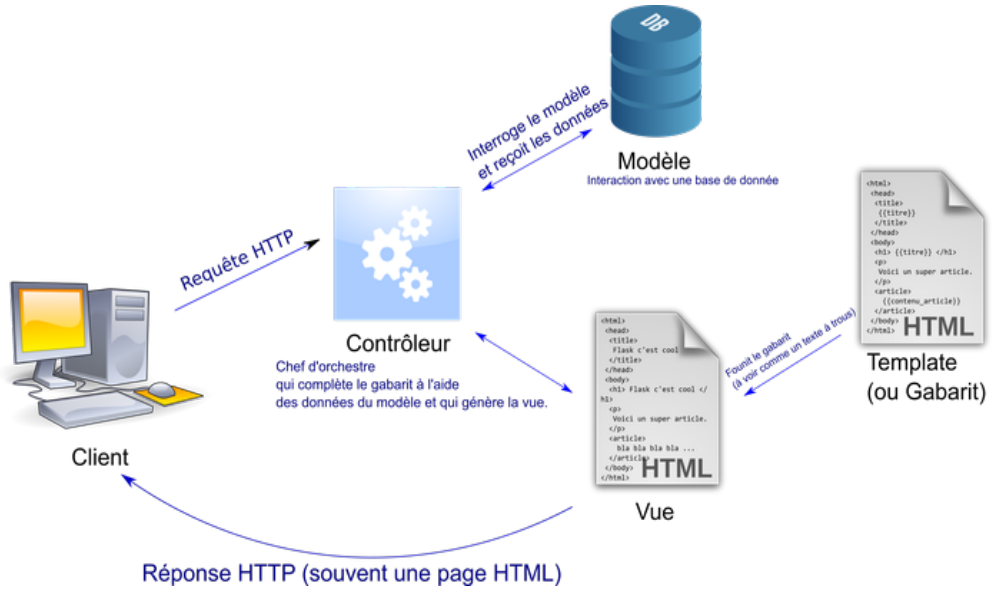

FIGURE 1 – fonctionnement de Flask

1.1 Un premier exemple

Le fichier principal d'un projet Flask contient au minimum les lignes suivantes :

```
from flask import Flask

app = Flask(__name__)

@app.route("/")
def root():
    return "<p>Hello world</p>"

app.run(debug=True)
```

Lors de son execution, une nouvelle tâche s'execute sur l'ordinateur *serveur*. Message en console :

```
Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Pour ouvrir la page envoyée par le serveur, il faut alors saisir l'URL donnée dans ce message.

1.2 Definition des termes app, vue, template et route

- app est une application Flask
- La fonction `root` est appelée une **vue**. Elle retourne une chaîne de caractères, qui sera le contenu de la réponse. Par défaut, le statut de la réponse est 200, et le type de contenu est HTML, encode en UTF-8.
- La ligne qui précède la fonction root, et commence par le symbole `@` est un décorateur python. Il sert à indiquer l'**URL** pour laquelle cette *vue* doit être utilisée.
- On appelle **route** une *URL* conduisant à l'exécution d'une fonction donnée.

1.3 Route avec paramètre

Une route peut être paramétrée, auquel cas le paramètre sera passé à la fonction vue :

```
@app.route("/hello/<name>")
def hello(name):
    return """<p>Hello %s</p>""" % name
```

Remarque : En python, le signe %s sert à substituer une variable de type string* par son contenu. (ici la variable *name*).

> Question : Quelle URL faut-il saisir pour afficher une page qui affiche le message : Hello Frank

1.4 templates

On peut vouloir construire une page HTML plus longue qu'un seul message de bienvenue. La *vue* doit alors recharger une page plus conséquente que l'on appelle *template*. Il faut alors utiliser la fonction `render_template` de la librairie Flask.

Exemple :

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

Supposons que le fichier principal, contenant les vues, s'appelle `main.py`. Les différentes pages, ou *templates*, seront alors mises dans un dossier appelé *templates* :

```
|- main.py
|- templates
        |- page1.html
```

Un template, ou "modèle", est un fichier dont certaines parties seront remplacées à l'exécution.

Voici un exemple minimaliste de template :

```
<p>Hello {{name}}</p>
```

Celui-ci contient en partie le script HTML, avec une partie entre doubles-accolades. Cette partie est écrite en langage **JINJA2**. Ici, il ne s'agit que d'une variable qui sera remplacée par son contenu lors de la construction de la page. Cette variable est elle-même contenue dans la *vue*.

Dans un template *Jinja2*, les doubles-accolades `{{ }}` servent à indiquer une substitution. Elles peuvent contenir un nom de variable, mais également des structures de contrôles (condition, boucle) similaires à celles de Python.

Elles sont encadrées par les symboles `{% %}`.

```
{% if elements %}
  <ul>
  {% for e in elements %}
    <li><a href="{{e.link}}">{{e.name}}</a></li>
  {% endfor %}
  </ul>
{% else %}
  <p>Aucun élément</p>
{% endif %}
```

> Question : On suppose que la variable `element` est une liste contenant 2 éléments, consitués de 2 liens et de 2 noms (voir tableau ci-dessous).

- Ecrire le script HTML généré par cet extrait en Jinja2.
- Représenter ce qui est affiché par le navigateur.

link	name
`www.lycos.fr`	lycos
`www.lequipe.fr`	journal l'équipe

Partie 2

Exemple complet : gestion d'un formulaire

Les vues Flask ne reçoivent pas directement l'information contenue dans la requête HTTP. Cette information est accessible via l'objet `flask.request`.

Cet objet possède un certain nombre d'attributs, dont : `method` généralement *GET* ou *POST*.

Voir seance de TP : Projet Flask