INF3005 – Programmation web avancée

Flask – partie 2

Jacques Berger

Objectifs

Connecter Flask à une BD

Utiliser un moteur de templating

Prérequis

Flask

Python

Sqlite

BD

Un serveur exécute plusieurs clients simultanément

Le serveur doit utiliser des connexions afin d'envoyer des requêtes à une BD

BD

Plusieurs clients ne peuvent utiliser une même connexion à la BD simultanément

Pour traiter plusieurs clients, il faut plusieurs connexions

BD

Établir une connexion à la BD est un processus coûteux

On utilise alors un pool de connexions

BD

Un pool de connexions est un ensemble de connexions à la BD disponibles qui attendent d'être utilisées

On prend une connexion dans le pool, on l'utilise, on la replace dans le pool une fois l'opération terminée

BD

Plusieurs librairies de BD implémentent un pool de connexions

Une transaction BD doit être complétée sur la même connexion

BD

Avec SQLite, comme il n'y a pas de communication réseau, on peut se permettre d'établir une nouvelle connexion pour chaque client

getattr

La fonction getattr permet d'obtenir une propriété d'un objet, sinon de retourner une valeur de défaut

BD

Voir exemple

BD

On tente d'isoler tout le code qui manipule directement une base de données (ex. requêtes SQL)

Si jamais on doit changer les données ou la BD, les changements seront plus faciles à identifier et à faire

Améliore la lisibilité et favorise le changement technologique

Conception

Lorsqu'on spécifie une nouvelle route, le code pour gérer la route devrait être court

Le code va gérer :

- Les données de la requête
- La réponse

Conception

S'il y a de la logique d'affaire à appliquer sur une route, la logique doit être déplacée dans une fonction distincte

kwargs

Permet de passer une série de propriétés/valeurs en paramètre à une fonction

```
une_fonction(param1='chaine', param2=3, param3='encore')
```

kwargs

La fonction reçoit les paramètres dans une variable : **kwargs

On accède aux propriétés/valeurs en faisant kwargs.items()

Render

La fonction render_template permet un kwargs comme deuxième paramètre

On peut ainsi envoyer de l'information à la template

Templating

Flask utilise le moteur de templating Jinja2

Un moteur de templating sert à générer du code HTML sans avoir à mélanger le code Python et le code HTML

Templating

Dans le template, on peut injecter une valeur texte reçue en paramètre

```
<p style="color:#700">{{ erreur }}</p>
```

Templating

Une template peut recevoir plusieurs données grâce au paramètre kwargs

```
return render_template('2listes.html', artists=artists,
                                       albums=albums)
```

Templating

On peut utiliser des conditions pour générer certaines sections de HTML

```
{% if erreur %}
  <p style="color:#700">{{ erreur }}</p>
{% endif %}
```

Templating

Pour boucler sur les éléments d'une liste :

```
{% for artist in artists %}
    <li>{{ artist }}</li>
{% endfor %}
```

Templating

Autre exemple de boucle :

```
<h1>Artistes</h1>
{% if not artists %}
  <p>Aucun artiste.</p>
{% else %}
  <ul>
    {% for artist in artists %}
        <li>{{ artist }}</li>
    {% endfor %}
  </ul>
{% endif %}
```

Héritage

Si on veut reproduire le même menu ou le même design d'une page à l'autre, on peut utiliser l'héritage de templates

Héritage

On définit le code qu'on veut avoir sur chaque page dans un template différent

Ce template sera le template de base

Il contient un bloc qui sera remplacé éventuellement

Héritage

Les autres templates vont hériter de la première et produire un document HTML représentant le bloc à remplacer dans le template de base

Voir exemple

Plus loin...

Using SQLite3 with Flask
    http://flask.pocoo.org/docs/0.12/patterns/sqlite3/

Jinja2
    http://jinja.pocoo.org/docs/2.9/