Flask - Introduction

Flask est un framework web léger en Python qui permet de développer des applications web de manière simple et rapide. Conçu pour être simple et extensible, Flask est un excellent choix pour les débutants en développement web. Dans cet article, nous allons découvrir comment installer Flask et créer une application web de base avec deux pages dynamiques en utilisant Jinja2.

5TTR Intérmédiaire

> Un **FRAMEWORK** est un ensemble d'outils, de bibliothèques et de bonnes pratiques qui facilitent le développement de logiciels en fournissant une structure et des fonctionnalités prêtes à l'emploi. Dans le contexte du développement web, un framework comme Flask fournit des outils pour gérer les **requêtes HTTP**, les **sessions**, les **templates HTML**, et bien plus encore, permettant aux développeurs de se concentrer sur la logique spécifique de leur application plutôt que sur les détails techniques de bas niveau.

Installation de Flask

Installation de Flask avec pip

Une fois que Python est installé, vous pouvez installer Flask en utilisant `pip`, le gestionnaire de paquets Python. Ouvrez un terminal et tapez la commande suivante :

```
pip install flask
```

Cela téléchargera et installera Flask ainsi que ses dépendances.

Dans Thonny IDE, allez dans le menu "Outils/Gérer les paquets" (GB ***Tools/Manage packages***) et cherchez le paquet nommé "Flask". Installez le package.

Création d'une Application Flask

Structure de base d'un projet Flask

Commençons par créer un répertoire `mon_projet_flask` pour notre projet. Ouvrez un terminal et tapez :

Ensuite, nous allons créer un fichier Python qui contiendra notre application Flask. Nommez ce fichier `app.py`.

Initialisation de l'application Flask

Ouvrez `app.py` dans votre éditeur de texte préféré et ajoutez le code suivant :

```python
from flask import Flask

app = Flask(__name__)

@app.route('/')
def home():
    return 'Bienvenue sur la page d\'accueil !'

if __name__ == '__main__':
    app.run(debug=True)
```

Ce code crée une instance de l'application Flask et définit une route pour la page d'accueil. La méthode `run` lance le serveur de développement.

Importation du module Flask

```python
from flask import Flask
```

Cette ligne importe le module Flask. Flask est un framework léger pour développer des applications web en Python. En important Flask, nous avons accès à toutes ses fonctionnalités et outils.

Création de l'application Flask

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

Ici, nous créons une instance de l'application Flask. L'objet `app` est l'application web que nous allons construire. `Flask(__name__)` indique à Flask de se configurer en fonction du module principal (notre fichier `app.py`).

Définition d'une Route

```python
@app.route('/')
def home():
    return 'Bienvenue sur la page d\'accueil !'
```

Cette partie du code définit une route pour notre application web. Une route est une URL spécifique associée à une fonction.

- `@app.route('/')` : Ce décorateur dit à Flask que cette fonction doit être appelée lorsque quelqu'un visite la racine du site web (l'URL `/`). Les décorateurs en Python sont une manière d'ajouter des fonctionnalités à une fonction existante.
- `def home():` : Nous définissons une fonction nommée `home` qui sera exécutée lorsque la route '/' est visitée.
- `return 'Bienvenue sur la page d\'accueil !'` : Cette ligne renvoie une chaîne de caractères à afficher sur la page web. Lorsque quelqu'un visite `http://localhost:5000/`, ils verront ce message.

> En termes simples, une **ROUTE** est une **association entre une URL (un chemin) et une fonction Python** dans votre application Flask. Cette fonction sera exécutée lorsque l'utilisateur accède à ce chemin dans son navigateur web. Autrement dit, **une "route" est un moyen de faire correspondre une URL (le chemin de la requête) à une fonction Python qui traitera cette requête**.

Lancement de l'application

```
if __name__ == '__main__':
    app.run(debug=True)
```

Cette section du code vérifie si le script est exécuté directement (et non importé comme un module). Si c'est le cas, elle lance le serveur de développement Flask.

- `if __name__ == '__main__':` : Cette ligne assure que le code qui suit ne s'exécute que si ce fichier est exécuté directement. Si le fichier est importé en tant que module dans un autre script, cette partie du code ne s'exécutera pas.
- `app.run(debug=True)` : Cette ligne démarre le serveur de développement de Flask. Le paramètre `debug=True` active le mode débogage, ce qui est utile pour le développement car il affiche les erreurs et recharge automatiquement l'application lorsque des modifications sont apportées.

> Une **ROUTE** (ou "chemin") est un terme utilisé en développement web pour décrire une URL spécifique dans une application web qui est associée à une fonction ou une méthode spécifique. Cette fonction est exécutée lorsque cette URL est accédée par un utilisateur. Dans Flask, par exemple, les routes sont définies pour associer des URL spécifiques à des fonctions Python, permettant de déterminer quelle partie du code doit être exécutée en fonction de l'URL demandée.:::

Lancement de l'application

Pour lancer l'application, retournez dans votre terminal et exécutez :

```
python app.py
```

Vous devriez voir un message indiquant que le serveur est en cours d'exécution, comme ceci :

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

Ouvrez un navigateur web et allez à l'adresse http://127.0.0.1:5000/. Vous devriez voir le message "Bienvenue sur la page d'accueil !".

Création de Pages Dynamiques avec Jinja2

Flask utilise Jinja2 comme moteur de templates pour générer des pages HTML dynamiques. Nous allons créer deux pages : une page d'accueil et une page "À propos".

Création des Templates

Créez un dossier nommé `templates` dans le répertoire de votre projet. À l'intérieur de ce dossier, créez deux fichiers : `home.html` et `about.html`.

home.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Accueil</title>
</head>
<body>
    <h1>Bienvenue sur la page d'accueil !</h1>
    <p>Ceci est une page dynamique générée avec Flask et Jinja2.</p>
    <a href="/about">À propos</a>
</body>
</html>
```

about.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>À propos</title>
</head>
<body>
    <h1>À propos de nous</h1>
    <p>Cette page présente des informations sur notre site.</p>
    <a href="/">Accueil</a>
</body>
</html>
```

Mise à jour de l'application Flask

Nous allons maintenant modifier `app.py` pour utiliser ces templates. Ouvrez `app.py` et mettez-le à jour comme suit :

```
from flask import Flask, render_template

app = Flask(__name__)

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

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

```

```python
if __name__ == '__main__':
    app.run(debug=True)
```

Dans ce code, nous utilisons la fonction `render_template` pour rendre les fichiers HTML que nous avons créés.

Test de l'application

Relancez l'application en exécutant à nouveau `python app.py` dans le terminal. Accédez à http://127.0.0.1:5000/ pour voir la page d'accueil et cliquez sur le lien "À propos" pour naviguer vers la page "À propos".

Conclusion

Dans cet article, nous avons installé Flask et créé une application web de base avec deux pages dynamiques en utilisant Jinja2. Flask est un outil puissant et flexible qui permet de développer des applications web rapidement. En explorant davantage Flask, vous découvrirez de nombreuses fonctionnalités supplémentaires pour créer des applications web robustes et complètes.

Continuez à expérimenter et à apprendre pour améliorer vos compétences en développement web avec Flask !

Vidéo à la une à regarder sur Youtube: http://youtu.be/lhp_cG7c2Rk