



Flask

Samuel SORIN
contact@samuelsorin.fr

Flask

Flask - c'est quoi ?

- Micro framework de développement web
- Flask a pour objectif de garder un noyau simple mais extensible
- Créé initialement par Armin Ronacher comme étant un poisson d'avril
- En 2018, Flask était élu "Framework web le plus populaire"
- Il se base sur deux modules werkzeug et jinja2

- <https://flask.palletsprojects.com/en/2.2.x/>
- [doc fr] <https://flask-fr.readthedocs.io/>

* werkzeug : Werkzeug est une bibliothèque d'applications Web WSGI complète

** WSGI : Web Server Gateway Interface. spécification qui décrit comment un serveur Web communique avec des applications Web

Flask - Fonctionnalités

- Serveur de développement et debugger
- Simplifie l'écriture de tests unitaires
- Moteur de template pour le rendu HTML (avec Jinja2)
- Supporte les cookies sécurisés (session)
- Entièrement compatible avec WSGI 1.0
- Se base sur l'Unicode
- Documentation complète
- Déploiement aisé sur plusieurs hébergeurs
- Ajout de fonctionnalités via les extensions

Flask - Première page (exercice exo1.py)

Il faut commencer par installer Flask.

```
pip install Flask
```

Puis créer un nouveau projet contenant un module python avec le code suivant :

```
from flask import Flask

app = Flask(__name__)

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

Pour lancer le serveur et afficher la page, vous devez saisir dans un terminal :

```
$ export FLASK_APP=mon_module.py           # ces lignes sont à saisir dans un terminal, dans le
dossier                                     # contenant votre module

$ export FLASK_ENV=development              # optionnel, permet de bénéficier du mode debug

$ flask run                                # lancement du serveur
```

Flask - Première page (exercice exo2.py)

Pour ne pas avoir à saisir les export dans un terminal, vous pouvez ajouter les dernières lignes à votre module puis exécuter directement le module :

```
from flask import Flask

app = Flask(__name__)

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

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

Flask - Utilisations des routes statiques (exercice exo3.py)

Il est aisé de créer plusieurs page avec une url spécifique pour chaque page.

Pour ce faire créer une nouvelle fonction qui définit la page précédée de `@app.route("/mon_url")`

```
from flask import Flask

app = Flask(__name__)

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

@app.route("/contact")
def contact():
    return """<h2>Contact</h2>
    <p>Bienvenue dans la page contact</p>"""

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

Flask - Utilisations des routes dynamique (exercice exo3.py)

Flask permet également d'ajouter simplement des paramètres aux routes, par exemple :

`@app.route('/user/<username>')` où `<username>` est le paramètre attendu dans la fonction qui suit

```
[...]

@app.route('/user/<username>')
def show_user_profile(username):
    return f'User {username}'

@app.route('/post/<int:post_id>')
def show_post(post_id):
    return f'Post {post_id}'

@app.route('/path/<path:subpath>')
def show_subpath(subpath):
    return f'Subpath {subpath}'

[...]
```


Flask - Bloquer l'accès des routes selon la méthode http utilisée

Flask permet de préciser le type de méthode (get, post, delete, put, patch...) autorisée pour accéder à une vue.

Particulièrement utile pour la création d'API

```
@app.route('/admin/articles/add/', methods=['GET', 'POST'])  
def articles_add():
```

Flask - Récupération des données POST/GET

Il est tout d'abord nécessaire de charger le module request de Flask :

```
from flask import request
```

Récupération des paramètres de l'url (get) :

```
title = request.args.get('title')
```

Récupération des données depuis un formulaire (post) :

```
title = request.form['title']          # génère une erreur si la clé n'existe pas  
title = request.form.get('name')      # renvoie None si la clé n'existe pas
```

Il est également possible de vérifier la méthode en cours d'utilisation :

```
if request.method == 'POST':  
    title = request.form['title']
```

Flask - Utilisé le moteur de template

Flask met à disposition le moteur de template Jinja2

```
from flask import Flask, render_template

@app.route('/articles/')
def article_list():
    items = get_articles()
    return render_template('article_list.html', articles=items)
```

render_template() accepte plusieurs paramètres.

- En première position le nom du template à utiliser. Celui-ci doit être placé dans un dossier 'templates/'
- ensuite il est possible d'ajouter autant de paramètres que souhaité. Ces paramètres seront disponibles dans le template. Ici les données "items" résultant de la fonction get_articles() seront accessible dans le template via la variable "articles"

Flask - plus d'infos

- [https://fr.wikipedia.org/wiki/Flask_\(framework\)](https://fr.wikipedia.org/wiki/Flask_(framework))
- <https://flask.palletsprojects.com/en/2.0.x/>

Flask

- exercice liste de course

- Reprendre l'application liste de course et la réécrire en projet Flask
- Fonctionnalités attendus :
 - une page unique générée avec jinja2
 - ajout, suppression d'item

Vous pouvez utiliser “app_skeleton” pour débiter le projet. (*app_skeleton met à disposition une structure minimale de projet Flask avec des fonctions utilitaires*)

Flask - exercice blog

- Reprendre le blog créé dans un précédent exercice et le réécrire avec Flask
- Fonctionnalités attendus :
 - Page liste des articles, avec un extrait de 40 mots seulement
 - Page détail d'un article, comptage du nombre de mot de l'article
 - Page administration avec listing des articles
 - Ajout, suppression, modification des articles dans l'admin
 - Titre des articles avec la première lettre en majuscule automatiquement

Vous pouvez “app_skeleton” pour débiter le projet. (*app_skeleton met à disposition une structure minimale de projet Flask avec des fonctions utilitaires*)