Introduction à Flask

Dev Web facile avec python

Sommaire

- Présentation de Flask
  - Pourquoi Flask ?
  - Les nombreuses extensions Flask
  - Flask plus en détails
- Anatomie d’un projet Flask

Présentation de Flask

Flask est un micro Framework de développement web écrit en python.

Allant à contre pied d'autres solutions web Flask est livré avec le strict minimum à savoir :

- un moteur de Template (Jinja 2)
- un moteur web de développement (Werkzeug)
- Un système de distribution de requête compatible REST (dit RESTful)
- Un support de débogage intégré au serveur web
- Un micro framework doté d'une très grande flexibilité
- Une très bonne documentation

Pourquoi Flask ?

Tout d'abord parce qu'il utilise **python** et que cela permet de garder une certaine homogénéité dans le traitement des données.

Cependant il existe d'autre alternative en python :

- Django
- Bottle
- CherryPy

Inconvénient :

- Nativement moins fournit que Django pour réaliser un développement de niveau professionnel (Authentification, gestion sécurisé, administration).

Avantage :

- Permet de produire les 1ère pages en moins de 5 min (installation comprise).
- Plus léger -> optimisation de ressource.
- Nombreuses extensions et populaire.

Les nombreuses extensions Flask

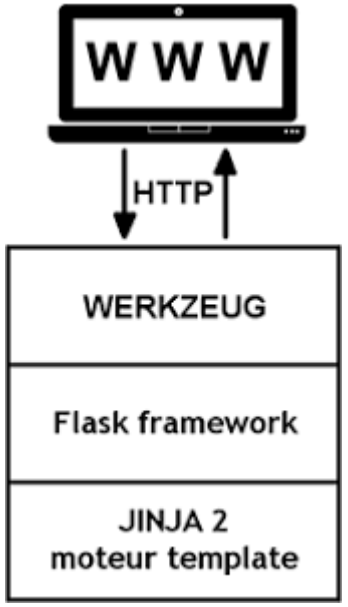
Flask dispose de très nombreuses extensions que l'on peut ajouter au cas par cas et dont la plupart sont installable avec l'utilitaire « pip » :

- Flask-Login : permet la gestion de sessions d'utilisateurs (login, déconnexion, mémorisation de session).
- Flask-User : gestion personnalisable des utilisateurs (enregistrement, confirmation, changement de mot de passe)
- Flask-Themes : permet de supporter plusieurs thèmes
- Flask-SQLAlchemy : apporte le support pour SQLAlchemy à Flask
- Flask-Mail : permet à une application Flask d'envoyer des e-mails
- …

Flask plus en détails

Les différents éléments composant Flask et intervenant dans la chaine de production de contenue :

- Werkzeug
- WSGI
- Application Flask
- Jinja
- La base de données



Flask plus en détails

La norme WSGI :

Le **Web Server Gateway Interface** et une norme et un protocole de communication qui définissent comment **un serveur web peut interagir avec un serveur python** pour envoyer des requêtes et recevoir des réponses.

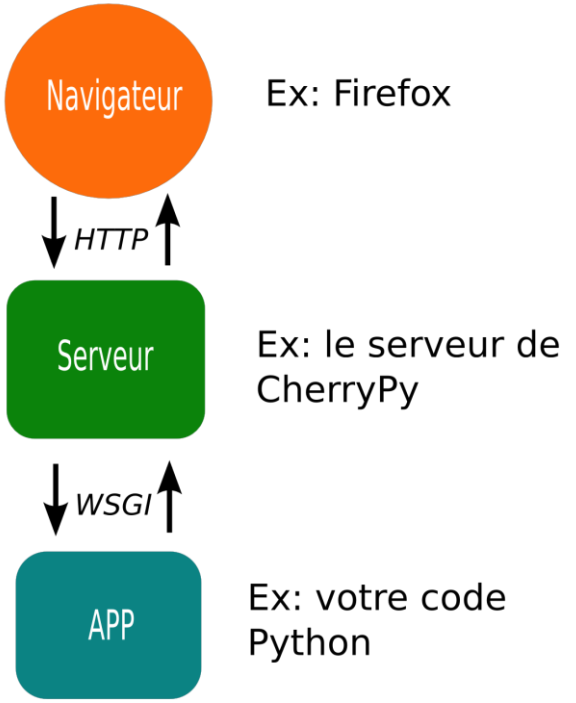
Un serveur web supportant WSGI doit donc être capable de transformer une requêtes HTTP en objet python et recevoir la réponse sous forme d'objet python.

Le but de cette norme : tous les serveurs compatible WSGI sont compatible avec toutes les applications compatible WSGI, ce qui assure que ces différents éléments sont interchangeables.

Flask plus en détails

Werkzeug :

- C'est un **serveur web WSGI** écrit en python.
  - *un serveur web (aussi appelé serveur HTTP) permet de répondre à une requête HTTP effectuée par un client (très souvent un navigateur web).*
- Il **reçoit les requêtes HTTP** et fait le nécessaire pour décoder les demandes et **envoyer les réponses en retour**. Il gère donc les connexions, mais prend aussi en charge des aspect sécuritaire au niveau HTTP.
- Il est capable de transmettre la requêtes à l'application en suivant le protocole WSGI, puis de récupérer la réponse et de la transmettre au client via HTTP.



Flask plus en détails

Application Flask :

- L'application utilisant le framework Flask **traite les requêtes entrantes et produit les réponses correspondantes**.
- L'application contient des « **routes »** permettant de réceptionner et de traiter les requêtes des clients pour produire les réponses correspondantes.
- Pour réaliser cette tâche l'application peut utiliser des **connexions vers une base de données** et obtenir les éléments nécessaire au rendu de la page.
- L'application peut directement produire du contenue HTML, CSV, JSON, XML et autre, ou alors s'appuyer sur le moteur de Template **Jinja**.

Flask plus en détails

Jinja :

- C’est un **moteur de Template** permettant de formater et d’inclure les données produite ou collecter par l’application dans les documents.
- Il est principalement utilisé pour produire du **contenu HTML**.
- Les documents HTML contiennent des **balises spéciales interprétées et remplacées** par le moteur de Template afin de produire le document définitif.
- **Jinja supporte le traitement de structures de contrôle** (test, boucle, assignation, comparaison, …), de filtres, l’héritage, les extensions, l’inclusion de block.

Flask plus en détails

La base de données :

- Le développeur **est libre de choisir la base de données** de son choix si celle-ci est disponible dans l'environnement python.
- Il est possible de faire appelle à un ORM (Object relation Mapper) tel que **SQLAlchemy** pour établir une connexion.
  - Jusqu'à présent nous avons utilisé SQLAlchemy seulement pour établir une connexion et Pandas pour aller écrire les données.

Anatomie d’un projet Flask

- L’élément le plus important d’un projet Flask est **la structure des répertoires** utilisée pour stocker les différents éléments du projet.
- Si la structure n’est pas **scrupuleusement respectée**, le projet ne produira pas de résultats.
- Ci-contre la structure de base du projet Flask pour le projet « mon-projet » avec quelques éléments complémentaires pour illustrer la mise en place.

```
mon-projet/
└── app
    ├── __init__.py
    ├── static
    │   ├── website.css
    │   ├── datagrid.css
    │   ├── ico-new.png
    │   ├── ico-top.png
    │   └── logo.png
    ├── templates
    │   ├── index.html
    │   ├── base.html
    │   └── entries.html
    ├── config.py
    ├── views.py
    └── ...
```

Anatomie d’un projet Flask

- **Fichier __init__.py** : Contient la création de l’objet applicatif Flask (qui sera appelé par le serveur Werkzeug). Application qui prendra en charge les différentes requêtes.
- **Répertoire static** : contient toutes les ressource static de l’application Flask tel que les images, script JavaScript, feuilles de styles, …
- **Répertoire templates** : contient les fichiers Template qui seront utilisés par Jinja pour produire les réponse renvoyé par l’application.
- **View.py** : Contient le traitement des requêtes (les vues) afin de les séparer du cœur de l’application. Ce fichier fait appelle au moteur de Template afin de renvoyer une réponse.