GrandPy Bot

Projet 7 - Développeur d'application – Python

Sommaire

- Démo
- Présentation / revue de code
- Q&A

Langages

- Python + Flask

- HTML

- CSS

Bootstrap

- Jquery

L'app Flask

Fichier : Views.py

- 2 context_processor → Affichage date & changement map (Google – OSM)
- 6 routes
  - 4 routes pour le site
    - /Index ou / → Index (homepage)
    - /go → Le dialogue Grandpy-Enfant
    - /about → Un a propos « musical »
    - /404 → Une page erreur 404
  - 2 routes « Ajax » :
    - /ajax → Traite la demande de renseignement de l'enfant
    - /goosm → Traite la modification du style de map à afficher

Les constantes et fichiers

Fichier : config.py

- URLS non terminées (manque paramètres et key)
- Messages aléatoires de Grandpy
- Style de map affichée (Google ou OSM)

Fichiers stopwords

- Répertoire /static/json
- 1 fichier récupéré sur Internet (stopwords.json)
- 1 fichier de mots personnels enrichi au fur et à mesure

Le front-end

Répertoires : /templates & /static

- Moteur de templates : Jinja2
- Navbar + footer
- Responsive → Bootstrap + css
- Scripts en jQuery
  - Appels Ajax
  - Modifications du DOM

Le back-end - Principe

Répertoires : /utils & views.py

- Routes gérés dans views.py
- Affichage dialogue : route /go
- Choix du style de map géré par route /goosm
  - Recharge page après pour charger la bonne librairie suivant choix
- Appel Ajax pour API google et wiki géré par route /ajax
  - Récupère la query telle qu'elle a été tapée au format JSON
  - Parse la query pour en sortir les mots clés
  - Fait appels aux apis Google et Wiki
  - Renvoi les réponse sous forme de données JSON

Le back-end – Le parser

Fichier : /utils/query.py

- 1 classe : Query
- Init → récupère la query + Build des noms des fichiers stopwords avec répertoires
- 1 méthode : parse_query → parser proprement dit
  - Supprime la ponctuation, les accents et les caractères « bizarres » de la query
  - Concatène les 2 fichiers stopwords et stopwordsperso
  - Pour chaque mot de la query le met dans une liste s'il n'est pas dans stopwords
  - Retourne la liste

Le back-end – les API - 1/3

Fichier : /utils/api.py

- **3 classes** :
  - Api → classe mère
    - Init → récupération de la query et mise en forme
    - api_get_json → Envoi de la requête → renvoi la réponse du requests.get
  - Gooapi → Api Google
  - Wikiapi → Api Wikipedia

Le back-end – les API - 2/3

Fichier : /utils/api.py

Gooapi (Google)

- Init
  - Build URL d'appel à l'API (query + key)

- get_json
  - Récupère le retour de Api.api_get_json
  - Vérifie le code retour : KO → renvoi 'Error'
  - Parse le retour (JSON)
  - Vérifie status : KO → renvoi 'Error'
  - Mets les valeurs intéressantes en liste (latitude, longitude…)
  - Renvoi la liste

Le back-end – les API - 3/3

Fichier : /utils/api.py

Wikiapi (Wikipedia)

- Init
  - Build URL d'appel à l'API ('name' retourné par Google)

- get_json : en 2 temps
  - Temps 1
    - Envoi requête parsée a l'API WIKI via Api.api_get_json
    - Récupération d'une pageid et d'un titre via API
  - Temps 2
    - Rebuild URL puis envoi a l'API Wiki d'une requete avec la pageid
    - Parse du retour et mise en liste
    - Renvoi de la liste

Les tests unitaires

Fichier : *test*/test_papybot.py

- Tests index et « not-found »
  - Création d'une fixture pytest pour simuler l'app et réaliser le get que ce soit en local ou sous Heroku
  - 2 tests qui vérifient le return-code HTTP (200 et 404)

- Test parser
  - Passe une phrase définie au parser qui doit renvoyer une autre valeur attendue

- Tests API
  - Utilisation des mocks pour les fonctions requests.get
  - 2 tests pour la bonne fin et 2 tests pour vérifier que les erreurs soient traitées