WEBCRAWLER

Projet 2

Université du Québec en Outaouais

Présenté au professeur Fraczak Wojciech

Dans le cadre du cours

Inf4533 technologies internet, groupe 01

16 avril 2017

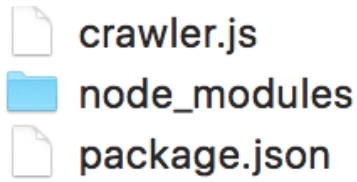
Martine pagé

PAGM03549702

WEBCRAWLER

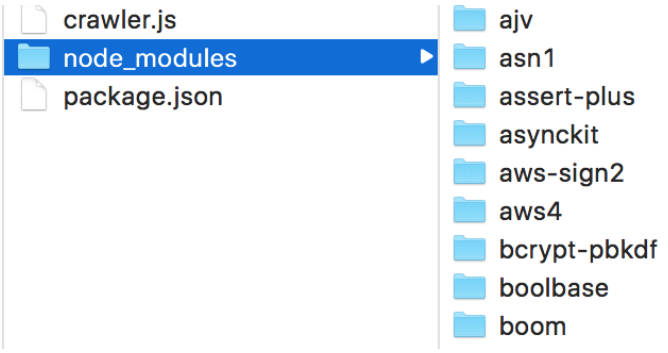
Dans le cadre du projet 2, j'ai tenté de faire un robot d'indexation pour que celui-ci recherche les événements ou exposition au Musée des Beaux Arts, au Musée de l'Histoire et au Musée de la Nature.

J'ai commencé par le programmer avant de faire une interface web. J'ai ensuite installé les logiciels prérequis dont Node.js et npm afin d'exécuter de code Javascript. Mon but premier était de sélectionner des pages web, dans ce cas il s'agissait des sites web des musées, et par la suite de sélectionner un mot afin de collectionner différents liens afin de pouvoir les visiter par la suite.



J'ai créé deux documents. Un document Javascript (crawler.js) pour la programmation générale et un document JSON (package.json) afin de stocker et échanger des données entre le navigateur et le serveur. Puisque Node contient une bibliothèque de serveur HTTP , il est alors possible d'avoir un serveur web sans avoir besoin d'un logiciel externe. Ensuite, j'ai installé la bibliothèque Node dans mon dossier afin d'y avoir accès en déterminant le lieu de téléchargement et en l'installant dans ce lieu déterminer.

```
MacBook-Pro-de-Martine:CRAWLER martinepage$ cd /Users/mar
MacBook-Pro-de-Martine:CRAWLER martinepage$ npm install
simple-webcrawler-javascript@0.0.0 /Users/martinepage/Des
├─┬ cheerio@0.19.0
│ ├─┬ css-select@1.0.0
│ │ ├── boolbase@1.0.0
│ │ ├── css-what@1.0.0
│ │ ├── domutils@1.4.3
│ │ └── nth-check@1.0.1
│ ├─┬ dom-serializer@0.1.0
│ │ └── domelementtype@1.1.3
│ ├── entities@1.1.1
│ ├─┬ htmlparser2@3.8.3
│ │ ├── domelementtype@1.3.0
│ │ └── domhandler@2.3.0
```



Les trois fichiers que j'ai utilisés de la bibliothèque s'agissent du Cheerio, Request, et URL. Request afin de faire des appels http, Cheerio pour extraire des données des pages Web et url pour briser un bloc de données en petits morceaux afin de les analyser par la suite.

C'est en utilisant le dossier Request qu'il est possible de visiter la page pour par la suite, faire un callback après avoir reçu la réponse. Et si, dans le terminal la réponse s'agit de 200, cela signifie que tout s'est bien passé.

```
[MacBook-Pro-de-Martine:~ martinepage$ cd /Users
 ebCrawler/Processus_WebCrawler
[MacBook-Pro-de-Martine:Processus_WebCrawler mar
 Visiting page http://www.beaux-arts.ca/fr/
 Status code: 200
 Found 121 relative links on page
 Visiting page http://www.beaux-arts.ca/fr/modal
 Status code: 200
Found 112 relative links on page
Visiting page http://www.beaux-arts.ca/fr/ress
Status code: 200
Found 126 relative links on page
Reached max limit of number of pages to visit.
MacBook-Pro-de-Martine:Processus_WebCrawler ma
```

SIMULATION WEBCRAWLER

Étant mes premières fois en programmation et en codage, après plusieurs tentatives, il m'était impossible d'appliquer ce code sur une page web pour que le robot d'indexation soit fonctionnel. C'est pourquoi j'ai décidé de simuler une Web Crawler. C'est-à-dire, j'ai créé une bibliothèque de 6 événements de la région en déterminant le nom, la date, la page web et en attribuant une image. Sur l'interface web, seulement 3 événements apparaissent aléatoirement afin de donner l'impression que les évènements se mettent à jour à chaque fois que la page est ouverte.