Marquer comme terminé

Initiation MongDB

## La quasi-intégralité des manipulations ci-dessous s'effectue sur la machine virtuelle utilisée depuis le début des TP.

- lancer la machine virtuelle utilisée pour faire les TP de `Postgres`
- s'y connecter en `ssh` depuis la machine hôte :

```
$ ssh -p 2222 tp@127.0.0.1
```

## Un premier contact avec **`MongoDB`**

- se connecter au serveur :

```
$ mongo
```

- obtenir la liste des databases :

```
> show databases
```

- créer une database

```
> use rioultf
```

- créer une collection en insérant un document :

```
> db.client.insert({nom: "toto"})
```

- obtenir un document dans la collection `client` :

```
> db.client.findOne()
```

- insérer un autre document :

```
> db.client.insert({nom: "titi"})
```

- obtenir tous les documentss :

```
> db.client.find()
```

- obtenir les documents dont le `nom` est `toto` :

```
> db.client.find({nom:"toto"})
```

- idem mais n'obtenir que l'identifiant :

```
> db.client.find({nom:"toto"}, {_id:1})
```

- supprimer la collection `client` :

```
> db.client.drop()
```

Pour faire du `MongoDB`, on va avoir besoin de manipuler des données JSON, ce que l'on va faire avec `jq`.

- télécharger les données d'exemple sur l'immobilier : https://ecampus.unicaen.fr/mod/resource/view.php?id=817061 ou directement à cette adresse : https://rioultf.users.greyc.fr/mongo/immo.json
- copier ces données de la machine hôte à la machine virtuelle :

```
$ scp -P 2222 immo.json tp@127.0.0.1:
```

## La suite des opérations s'effectue sur la machine virtuelle

## Initiation **`jq`**

Expliquer ce que font les commandes `jq` suivantes (vous pouvez utiliser https://jqplay.org/) :

?

```
$ jq . immo.json
$ jq .[] immo.json
$ jq .[1] immo.json
$ jq .[2,4] immo.json
$ jq .[].id immo.json
$ jq .[].owner immo.json
$ jq -c .[].owner immo.json
$ jq .[].owner.firstName immo.json
```

Avec `jq`, on peut enchaîner et imbriquer les filtres :

```
$ jq '.[] | select(.owner.firstName == "Kirk")' immo.json
$ jq '.[] | select(.owner.firstName == "Kirk") | {id, price}' immo.json
$ jq '.[] | select(.owner.firstName | test("^Jo")) | {owner}' immo.json
$ jq '.[1] | keys' immo.json
```

## Pour aller plus loin avec `jq` :

- Le manuel complet est à https://stedolan.github.io/jq/manual/
- un bon post avec beaucoup de détails, entre autres de la conversion CSV -> JSON : https://programminghistorian.org/en/lessons/json-and-jq

## Insertion des données dans MongoDB

L'import de données s'effectue au format BSON : il s'agit de mettre dans un fichier (ou un flux) *un document par ligne*. `jq` s'acquitte parfaitement de cette tâche avec l'option `-c`.

Il suffit ensuite d'envoyer le flux dans `mongoimport` en précisant la base de données (option `-d`) et la collection (option `-c`) :

```
    $ jq -c .[] immo.json | mongoimport -d immo -c biens
```

## Quelques requêtes pour s'entraîner avec Mongo :

Ces requêtes sont à écrire avec `Mongo`, pas avec `jq` !

- afficher uniquement les identifiants des biens
- afficher les biens dont le prix est supérieur à 500 000
- ceux dont le prix est entre 500 et 800 000
- ceux dont le prix est en-dessous de 500 000 ou au-dessus de 800 000
- ceux dont le propriétaire a pour prénom `Joyce`
- ceux dont le propriétaire a un prénom qui commence par `J` et un mail en `.biz`

Modifié le: jeudi 3 octobre 2024, 16:32

◄ Données de corpus

Choisir un élément

Aller à…

TP 6.2 ►

mentions légales . vie privée . charte utilisation . unicaen . cemu . moodle