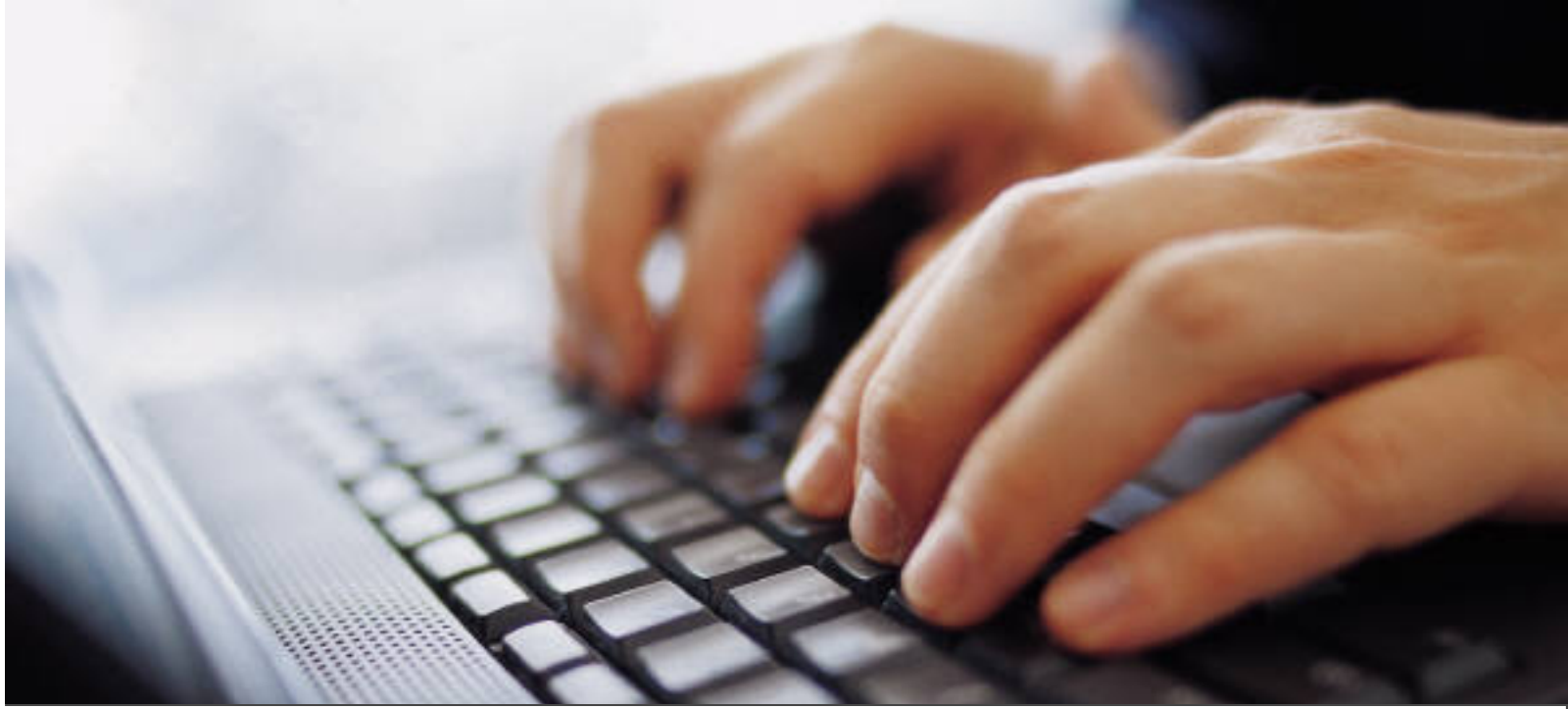
Hadoop / Big Data

Benjamin Renaut <renaut.benjamin@tokidev.fr>

MBDS

2019 - 2020



TP 3

MongoDB – Utilisation et API Java

Installation MongoDB

- **Pour ce TP, la même machine virtuelle que pour le TP précédent va être utilisée. Lancez la machine virtuelle. Connectez-vous avec l'utilisateur mbds.**

- **Démarrez MongoDB avec la commande:**
  `start-mongodb`

- **Une MongoDB démarré, vous serez en mesure de vous connecter à la base directement depuis votre machine hôte du moment que la machine virtuelle fonctionne (est lancée); une redirection de port a été ajoutée à la machine virtuelle pour le permettre.**

- **Cela suppose que le port TCP MongoDB (27017) ne soit pas d'ores et déjà utilisé sur votre machine par un autre serveur MongoDB (ou autre); si c'est le cas, arrêtez votre instance MongoDB locale.**

Importation des données

- **Lancez ensuite les commandes suivantes pour importer nos données de test:**

```
wget http://cours.tokidev.fr/bigdata/tps/primer-mongodb.json.gz
gunzip primer-mongodb.json.gz
mongoimport --db data --collection restaurants --drop --file primer-mongodb.json
```

- **Si l'import s'est correctement réalisé, vous devriez obtenir un message:**

```
25359 document(s) imported successfully
```

Préparation du TP

- **Vous pouvez désormais lancer le *shell* MongoDB:**

```
mongo
```

- **A noter que vous devriez également pouvoir le lancer sur la machine hôte si le shell mongoDB est installé sur votre machine, grâce au *port forwarding* mis en place.**

- **Ceci vous permettra également de compiler et d'exécuter votre code sur votre machine hôte directement si vous le souhaitez dans la partie Java.**

Partie 1 – Utilisation du *shell*

- **L'instance MongoDB comporte désormais une base de données de test, « data ».**
- **Cette base de données comporte une collection « restaurants ».**
- **La première partie du TP consiste à effectuer une série d'opérations sur cette collection pour se familiariser avec le *shell* MongoDB.**
- **Commencez par observer le format des différents documents présents dans la collection pour vous familiariser avec ce format.**
- **La première étape consiste donc à sélectionner des enregistrements de la collection restaurants.**

Partie 1 – Utilisation du *shell*

- **Essayez maintenant de:**
  - **Sélectionner tous les restaurants dont le type de cuisine est italienne.**
  - **Sélectionner tous les restaurants italiens dont le code postal est "10075".**
  - **Sélectionner tous les restaurants dont le type de cuisine est italienne OU américaine.**
  - **Sélectionner tous les restaurants italiens dont la note (le score) est supérieur à 50, en les triant par nom de restaurant et en n'en gardant que les 10 premiers, après avoir sauté les 5 premiers.**
  - **Mettre à jour tous les enregistrements dont le type de cuisine est "Other" pour qu'il devienne "Autre".**

Partie 1 – Utilisation du *shell*

- **Insérer un nouveau restaurant. C'est vous qui choisissez le contenu des données; mais essayez de respecter le format général existant.**

- **Mettre à jour votre enregistrement pour changer son type de cuisine.**

- **Vous pouvez évidemment vous entraîner à effectuer d'autres opérations pour vous familiariser avec le *shell*.**

Partie 2 – API Java

- **Vous devez désormais réaliser un petit programme Java (classique) qui a pour but de gérer un annuaire.**

- **Vous devez à minima supporter les champs: nom, prenom, et adresse e-mail.**

- **Votre programme doit permettre de:**
  - **Chercher un contact (par son adresse e-mail ou le couple nom & prénom, par exemple).**
  - **En ajouter un.**
  - **En supprimer un.**

- **Vous pouvez au choix permettre ces opérations via les arguments de la ligne de commande (par exemple `-add nom,prenom,email`) ou *via* une interface de lecture clavier. Le mode texte est dans tous les cas recommandé (ne pas perdre de temps à créer un GUI).**

Partie 2 – API Java

- **Vous devez créer une nouvelle base de données et une nouvelle collection pour stocker vos données. C'est vous qui choisissez le format de données.**

- **Si vous avez fini en avance, essayez de:**

  - **Rajouter la possibilité de modifier un contact.**
  - **Vérifier qu'un document a bien été modifié aprés la modification d'un contact; et en afficher une confirmation.**

- **Pour cette dernière fonctionnalité, pensez à la méthode getModifiedCount de l'objet UpdateResult mentionné en cours !**

- **Vous aurez par ailleurs besoin de la librairie (API MongoDB): vous pouvez utiliser Maven ou Gradle pour l'obtenir plus efficacement (voir ici: https://mongodb.github.io/mongo-java-driver/)**