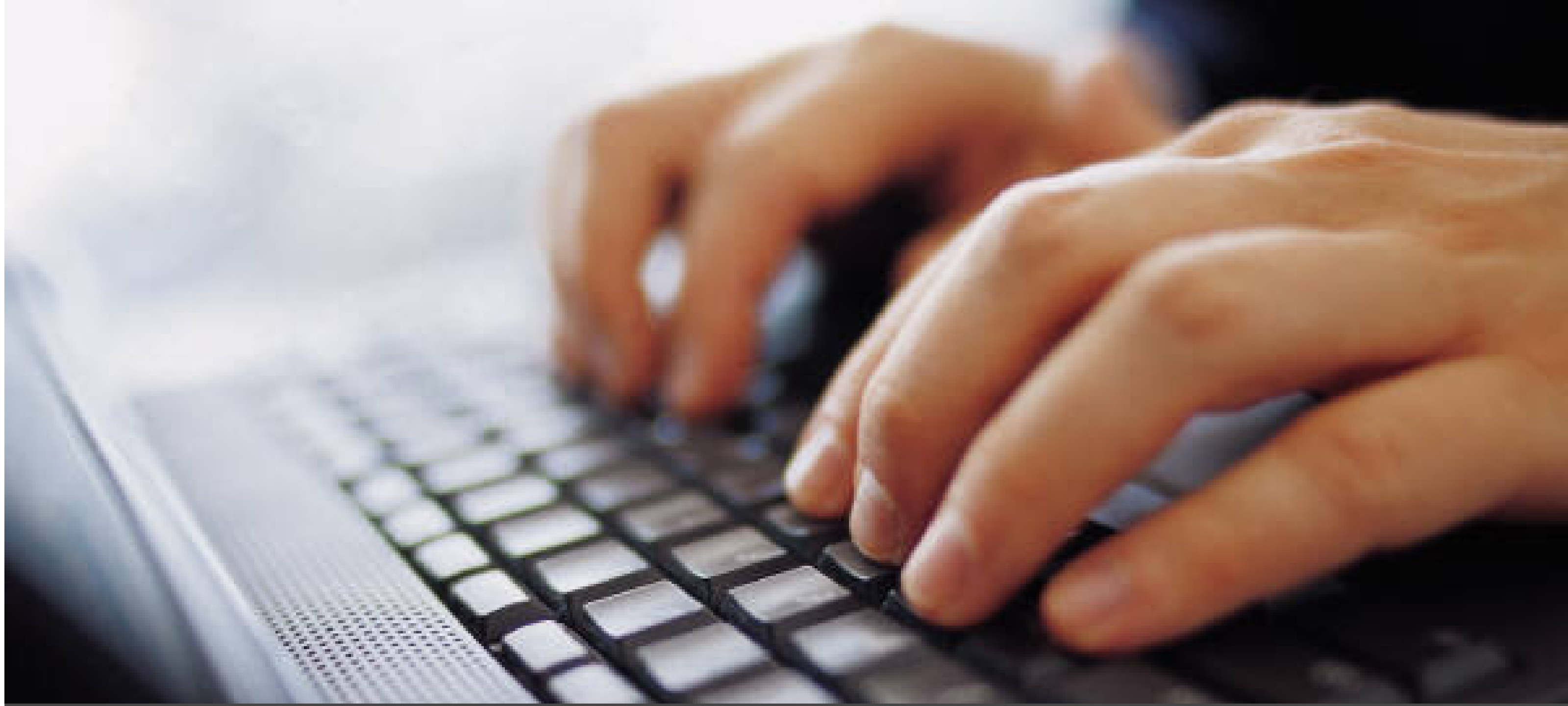
Hadoop / Big Data

Benjamin Renaut <renaut.benjamin@tokidev.fr>

MBDS

2019 - 2020



TP 3

MongoDB – Usage and Java API

Setup

- For this exercise, the same virtual machine that was used previously will again be used. Start it, then login using the « mbds » user.

- Start MongoDB with the command:
  ```
  start-mongodb
  ```

- Once MongoDB has started, you will be able to connect directly to it from your host machine; a port forwarding has been setup so that this is possible.

- This of course assumes the MongoDB TCP port (27017) isn’t already being used by another MongoDB server (or another service) on your host machine; if the port is being used, stop your host MongoDB server or modify the port forwarding setup in Virtual Box.

Importing the example data

- Use the following commands to import our test data:

```
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
```

- If the import succeeded, you should see a message saying:

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

Setup

- You can now start the MongoDB shell:

```
mongo
```

- Remark: you should also be able to run mongo directly on your own machine if you have the MongoDB shell installed here (thanks to the port forwarding). This will also allow you to directly connect to the server if needed directly from your host machine (allowing you to « run » your program directly from the IDE).

Part 1 – MongoDB shell usage

- **The MongoDB server now has a test database, « data ».**
- **This database contains a collection, « restaurants ».**
- **The first part of this exercise is to execute a series of operations using the MongoDB shell; allowing you to familiarize yourself with it.**
- **Start by checking out the format of the documents in the collection.**
- **Your first step will therefore be to select all documents in the restaurants collection.**

Part 1 – MongoDB shell usage

- Now try to:
  - Select all restaurants for which the cuisine type is « Italian ».
  - Select all Italian restaurants for which the postcode is "10075".
  - Select all restaurants that are either of type « Italian » or « American ».
  - Select all Italian restaurants for which the note (score) is higher than 50, sorting them by restaurant name and keeping only the first 10, after skipping the first 5.
  - Update all restaurants for which the type is defined as « Other » so that it now becomes « Misc ».

Part 1 – MongoDB shell usage

- Insert a new restaurant. You can use the format you wish; but try to respect the existing general format.

- Update your own, new restaurant record so that its cuisine type becomes « Special ».

- You can of course try and attempt other operations to familiarize yourself with the MongoDB shell.

Part 2 – Java API

- **You should now develop a simple, classical Java program to access MongoDB: a contact directory.**

- **You should at least support the following fields for a contact: name, surname, and email.**

- **Your program should allow to:**
  - **Search a contact (by mail or name, for example).**
  - **Add a contact.**
  - **Delete a contact.**

- **You can either allow specifying the operation using the command line parameters (for example `-add name,surname,email`) or through an interactive keyboard interface. It is in any case recommended not to develop a true graphical UI; use the console.**

Part 2 – Java API

- You should create a new database and a new collection to store your contact directory data.

- If you've finished early, try to:

  - Allow to update an existing contact.
  - Add a feature to check and make sure a contact has indeed been modified in the databse after being requested, and displaying a confirmation.

- For that last feature, consider the getModifiedCount method of the UpdateResult object described in the course material.

- Note that you will required a Java library (MongoDB API): you can use Gradle or Maven to fetch it easily (see here: https://mongodb.github.io/mongo-java-driver/)