MongoDb et Java

Introduction

Pour attaquer MongoDb depuis Java, on utilise le driver **mongo-java-driver-*version*.jar** (ici, version vaut 3.6.4).

Pour pouvoir effectuer des requêtes, il faut créer un objet *MongoClient*. Ex :

```
MongoClient mclient = new MongoClient("localhost");
```

(Pour donner une idée, le *MongoClient* joue le rôle d'une *Connection* dans une base SGBDR)

Dans la suite du texte, on utilisera indifféremment des *Document* ou des *Map<String,Object>*, les deux étant quasi-identiques (pour être précis, *Document* implémente l'interface *Map*).

L'Id

Chaque Document possède un id unique qui est créé automatiquement. Cet Id est accessible via le champ *_id* du *Document*. Contrairement aux bases SGBDR, cet Id n'est pas un long. C'est un objet particulier, *ObjectId* qui peut être représenté sous forme de chaine de caractères.

Création des Bases et Collections

Une des beautés de MongoDb, c'est qu'il n'est pas nécessaire de créer explicitement les databases et les collections : il suffit de les utiliser pour qu'elles soient créées si elles n'existent pas.

```
MongoDatabase database = mclient.getDatabase(dbName);
MongoCollection<Document> docs = database.getCollection(collectionName);
```

Ici, les objets *database* (base de données) et *docs* (Collection) sont automatiquement créées.

Récupération d'un document

Pour Récupérer un document (pour un get, un delete, un update), il faut repérer le document en utilisant un find() …

Exemple :

```
Document doc = null;
BasicDBObject search = new BasicDBObject();
search.put("_id", new ObjectId(id));
MongoCursor<Document> cur = docs.find(search).iterator();
while (cur.hasNext()) {
	doc = cur.next();
	break;
}
```

Ici on recherche un Document par son id :

- On crée un objet de recherche *BasicDBObject*
- On spécifie que cet objet va rechercher sur le id (*_id*) et on lui passe *id* qui contient l'id suivant lequel on va rechercher.
- On créer un *MongoCursor* pour récupérer tous les Document de *docs* (Collection) dont le _id vaut id.
- On récupère le Document (s'il y en a un)

Lister

C'est un peu pareil :

- On crée éventuellement un *BasicDBObject* pour faire une recherche. Si on n'en créé pas, on renverra tous les *Document* de la collection
- On utilise un *MongoCursor* et on boucle pour récupérer une liste de Document

Détruire

Toujours la même chose :

- On recherche un ou des éléments (par exemple suivant l'id)
- On le détruit

```
BasicDBObject search = new BasicDBObject();
search.put("_id",new ObjectId(id));
docs.deleteOne(search);
```

Créer

```
Map<String,Object> map = new HashMap<>() ;
…
… On charge la Map
…
Document doc = new Document(map);
docs.insertOne(doc);
map.put("_id", doc.get("_id")); // on met l'id créé dans la map
```

Update

Là, c'est un peu plus compliqué. Il faut récupérer l'objet (ou les objets), et le/les mettre à jour en utilisant le *$set*.

```
Map<String,Object> map = … // on récupère un Document via son _id (voir + haut)
Document newDoc = new Document(newMap); // newMap a été chargée en amont
docs.updateOne(new Document(map),new Document("$set", newDoc));
```