



Université Sultan Moulay Slimane  
École Nationale des Sciences Appliquées  
-Khouribga-



Filière : Génie Informatique

---

# Rapport TP MongoDB avec Java

---

*Réalisé par :*

Mohammed Reda KADIRI

*Encadré par :*

Pr. Nassima SOUSSI

Année universitaire  
2024/2025

# Chapitre 1

## Connexion à MongoDB

Dans cette section, nous allons expliquer comment établir une connexion à la base de données MongoDB en utilisant le pattern Singleton.

### 1.1 Question 1 : Afficher toutes les bases de données existantes.

Le pattern Singleton permet de s'assurer qu'une seule instance de la connexion à la base de données est créée et utilisée dans toute l'application. Cela optimise l'utilisation des ressources et évite la surcharge due à la création de multiples connexions.

```
1 package org.example;
2
3 import com.mongodb.client.MongoClient;
4 import com.mongodb.client.MongoClients;
5
6 class MongoDBConnection { 5 usages
7     private static MongoDBConnection instance; 3 usages
8     private final MongoClient mongoClient; 2 usages
9     private static final String url = "mongodb://localhost:27017"; 1 usage
10
11     private MongoDBConnection() { 1 usage
12         mongoClient = MongoClients.create(url);
13     }
14
15     public static synchronized MongoDBConnection getInstance() { 1 usage
16         if (instance == null) {
17             instance = new MongoDBConnection();
18         }
19         return instance;
20     }
21
22     public MongoClient getConnexion() { 12 usages
23         return mongoClient;
24     }
25 }
```

FIGURE 1.1 – Illustration du pattern Singleton

### 1.1.1 Explication du code

Le fichier Java correspondant contient une classe Singleton qui :

- Initialise une seule instance de la connexion à MongoDB.
- Fournit une méthode statique pour récupérer cette instance.
- Empêche l'instanciation multiple de la connexion.

Cette approche garantit une gestion efficace des ressources et évite les problèmes liés aux connexions multiples inutiles à la base de données MongoDB.

### 1.1.2 Code de Methode qui liste les databases :

code de la methode *listeDatabases()*

```
public void updateDocument(String collectionName, String name, String newEmail) { no usages
    MongoDB database = ConnectionInstance.getConnexion().getDatabase( "NewDatabase");
    MongoCollection<Document> collection = database.getCollection(collectionName);

    // Filtrer le document par nom
    Document filter = new Document("name", name);

    // Mettre à jour le champ "email" du document trouvé
    Document update = new Document("$set", new Document("email", newEmail));

    try {
        // Mettre à jour le premier document qui correspond au filtre
        collection.updateOne(filter, update);

        System.out.println("Le document avec le nom " + name + " a été mis à jour avec le nouvel email : " + newEmail);
    } catch (Exception e) {
        System.err.println("Erreur lors de la mise à jour du document : " + e.getMessage());
    }
}
```

FIGURE 1.2 – code de ListeDatabases()

la Classe de Test

```

package org.example;

import ...

public class TestClass {
    public static void main(String[] args) {
        Methodes methodes = new Methodes();
        Document doc = new Document("name", "John Doe")
            .append("age", 29)
            .append("email", "johndoe@example.com")
            .append("address", new Document("street", "123 Main St")
                .append("city", "Somewhere")
                .append("postalCode", "12345"));
        methodes.listDatabases();
        //methodes.listCollections("gi2");
        //methodes.CreateDatabase("newDatabase");
        //methodes.DeleteDatabase("newDatabase");
        //methodes.CreateCollection("Second_Collection_test" );
        //methodes.ShowCollection("Second_Collection_test");
        //methodes.DeleteCollection("First_Collection_test");
        //methodes.addDocumentToCollection(doc,"First_Collection_test");
        //methodes.ShowDocument_Of_Collection("John Doe", "First_Collection_test");
        //methodes.deleteDocument("First_Collection_test", "redaa");
    }
}

```

FIGURE 1.3 – Class Test

Resultat :

```

"C:\Program Files\Java\jdk-23\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\
Mar 19, 2025 1:56:33 AM com.mongodb.internal.diagnostics.logging.Loggers show
WARNING: SLF4J not found on the classpath. Logging is disabled for the 'org.mo
Base de donnees: NewDatabase
Base de donnees: admin
Base de donnees: config
Base de donnees: gi2
Base de donnees: library
Base de donnees: local

Process finished with exit code 0

```

FIGURE 1.4 – Resultat Question 1

### 1.1.3 Question 2 : Créer/supprimer une base de données.

```

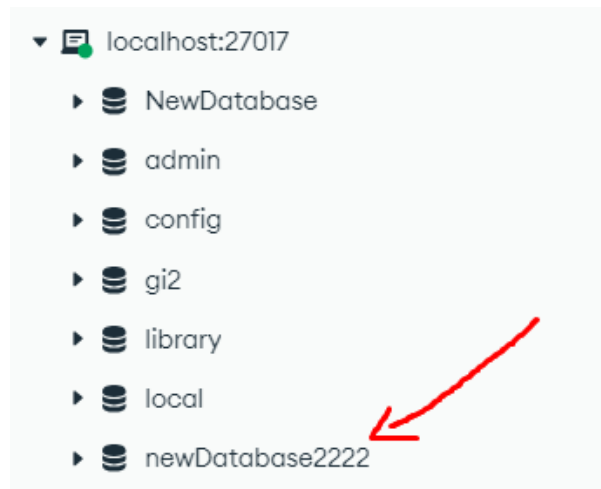
public void CreateDatabase(String dbname){ no usages
    MongoDB database = ConnectionInstance.getConnexion().getDatabase(dbname);
    database.createCollection( "s: "Default_Collection");
    System.out.println("The new database created is: " + dbname );
}

```

FIGURE 1.5 – creation base de donnee

Resultat :

```
"C:\Program Files\Java\jdk-23\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ
Mar 19, 2025 2:10:10 AM com.mongodb.internal.diagnostics.logging.Loggers shouldUse
WARNING: SLF4J not found on the classpath. Logging is disabled for the 'org.mongodb
The new database created is: newDatabase2222
```



Supprimer un Base de donnee

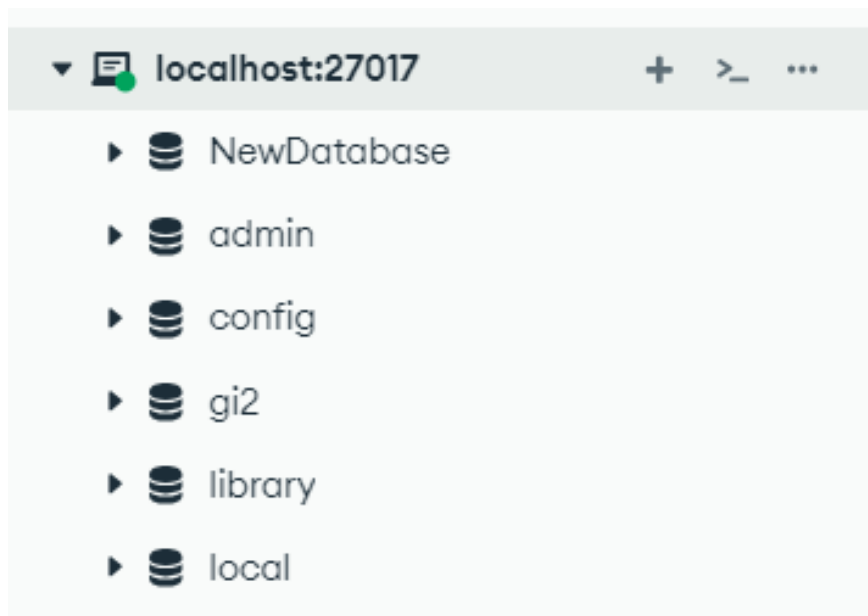
```
public void updateDocument(String collectionName, String name, String newEmail) { no usages
    MongoDBDatabase database = ConnectionInstance.getConnection().getDatabase( s: "NewDatabase");
    MongoCollection<Document> collection = database.getCollection(collectionName);

    // Filtrer le document par nom
    Document filter = new Document("name", name);

    // Mettre à jour le champ "email" du document trouvé
    Document update = new Document("$set", new Document("email", newEmail));

    try {
        // Mettre à jour le premier document qui correspond au filtre
        collection.updateOne(filter, update);

        System.out.println("Le document avec le nom " + name + " a été mis à jour avec le nouvel email : " + newEmail);
    } catch (Exception e) {
        System.err.println("Erreur lors de la mise à jour du document : " + e.getMessage());
    }
}
```



### 1.1.4 Question 3 : Afficher toutes collections existantes d'une BD.

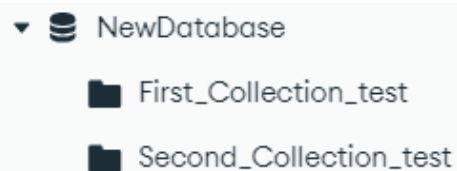
```
public void listCollections(String dbName) { no usages
    MongoDBDatabase database = ConnectionInstance.getConnexion().getDatabase(dbName);
    System.out.println("Collections dans la base de données '" + dbName + "' :");
    for (String collectionName : database.listCollectionNames()) {
        System.out.println("- " + collectionName);
    }
}
```

```
WARNING: GET4S not found on the classpath.
Collections dans la base de données 'gi2' :
- tweets
- livres
- statistique
```

### 1.1.5 Question4 : Créer/afficher/supprimer une collection

Creation d'une collection :

```
public void CreateCollection(String Collection){ no usages
    MongoDBDatabase database = ConnectionInstance.getConnexion().getDatabase(s: "NewDatabase");
    database.createCollection(Collection);
    System.out.println("the collection Created is: "+ Collection );
}
```



```
▼ [Database Icon] NewDatabase
    [Folder Icon] First_Collection_test
    [Folder Icon] Second_Collection_test
```

## Affichage d'une collection :

```
public FindIterable<Document> ShowCollection(String CollectionName){ no usages
    MongoDB database = ConnectionInstance.getConnexion().getDatabase( s: "NewDatabase");
    MongoCollection<Document> collection = database.getCollection(CollectionName) ;
    FindIterable<Document> documents_de_collection = collection.find();
    Iterator<Document> it = documents_de_collection.iterator();
    System.out.println("The Collection selected is: " + CollectionName);
    while (it.hasNext()) {
        System.out.println(it.next().toJson()); //on peut enlever json
    }
    return documents_de_collection;
}
```

```
WARNING: SLF4J not found on the classpath. Logging is disabled for the 'org.mongodb.driver' component
The Collection selected is: Second_Collection_test
{"_id": {"$oid": "67d6070ef5f04b85d3abaa08"}, "fruit": "Apple", "size": "Large", "color": "Red"}
```

## Supression d'une collection :

```
public void DeleteCollection(String CollectionName){ no usages
    MongoDB database = ConnectionInstance.getConnexion().getDatabase( s: "NewDatabase");
    MongoCollection<Document> collection = database.getCollection(CollectionName) ;

    try {
        if(Collection_Exist(CollectionName)){
            collection.drop();
            System.out.println("The collection '" + CollectionName + "' has been deleted.");
        }else{
            System.err.println("Error: The collection '" + CollectionName + "' does not exist.");
        }
    } catch (RuntimeException e) {
        System.err.println("Error deleting collection '" + CollectionName + "': " + e.getMessage());
    }
}
```

```
WARNING: SLF4J not found on the classpath. Logging is disabled
The collection 'First_Collection_test' has been deleted.
```



### 1.1.6 Question 5 : Ajouter un document à une collection

```
public void addDocumentToCollection(Document document, String CollectionName){ no usages
    MongoDB database = ConnectionInstance.getConnexion().getDatabase( s: "NewDatabase");
    MongoCollection<Document> collection = database.getCollection(CollectionName);

    try{
        collection.insertOne(document);
        System.out.println("The document has been added to the collection '" + CollectionName + "'.");
    }catch (RuntimeException e){
        System.err.println("Error adding document to collection '" + CollectionName + "': " + e.getMessage());
    }
}
```

WARNING: SLF4J not found on the classpath. Logging is disabled for the 'org.mongodb.internal.diagnostics.logging.Loggers'.

The document has been added to the collection 'Second\_Collection\_test'.





### 1.1.7 Question 6 : Afficher/mettre à jour/supprimer un document bien déterminé

Afficher un document bien déterminé :

```
public void ShowDocument_Of_Collection(String DocName, String CollectionName){ no usages
    MongoDB database = ConnectionInstance.getConnexion().getDatabase( s: "NewDatabase");
    MongoCollection<Document> collection = database.getCollection(CollectionName);

    Document filter = new Document("name", DocName);
    Document document = collection.find(filter).first(); // .first() renvoie le premier résultat ou null
    if (document != null) {
        // Afficher le document trouvé
        System.out.println("Document trouvé : " + document.toJson());
    } else {
        System.out.println("Aucun document trouvé avec le nom '" + DocName + "'");
    }
}
```

FIGURE 1.6 – Methode qui affiche un document

on va afficher le document qui a comme propriete Name : "John Doe" car on a creer un filter sur cette Propriete **Name** :

```
public void updateDocument(String collectionName, String name, String newEmail) { no usages
    MongoDB database = ConnectionInstance.getConnexion().getDatabase( s: "NewDatabase");
    MongoCollection<Document> collection = database.getCollection(collectionName);

    // Filtrer le document par nom
    Document filter = new Document("name", name);

    // Mettre à jour le champ "email" du document trouvé
    Document update = new Document("$set", new Document("email", newEmail));

    try {
        // Mettre à jour le premier document qui correspond au filtre
        collection.updateOne(filter, update);

        System.out.println("Le document avec le nom '" + name + "' a été mis à jour avec le nouvel email : " + newEmail);
    } catch (Exception e) {
        System.err.println("Erreur lors de la mise à jour du document : " + e.getMessage());
    }
}
```

Supprimer un document bien déterminé :

```
public void deleteDocument(String collectionName, String name) { 1 usage
    MongoDBDatabase database = ConnectionInstance.getConnexion().getDatabase(s: "NewDatabase");
    MongoCollection<Document> collection = database.getCollection(collectionName);

    Document filter = new Document("name", name);
    try {
        collection.deleteOne(filter);
        System.out.println("Le document avec le nom " + name + " a été supprimé.");
    } catch (Exception e) {
        System.err.println("Erreur lors de la suppression du document : " + e.getMessage());
    }
}
```

WARNING: SLF4J not found on the classpath. Logging is disabled.  
Le document avec le nom 'John Doe' a été supprimé.

```
public void updateDocument(String collectionName, String name, String newEmail) { no usages
    MongoDBDatabase database = ConnectionInstance.getConnexion().getDatabase(s: "NewDatabase");
    MongoCollection<Document> collection = database.getCollection(collectionName);

    // Filtrer le document par nom
    Document filter = new Document("name", name);

    // Mettre à jour le champ "email" du document trouvé
    Document update = new Document("$set", new Document("email", newEmail));

    try {
        // Mettre à jour le premier document qui correspond au filtre
        collection.updateOne(filter, update);

        System.out.println("Le document avec le nom " + name + " a été mis à jour avec le nouvel email : " + newEmail);
    } catch (Exception e) {
        System.err.println("Erreur lors de la mise à jour du document : " + e.getMessage());
    }
}
```

FIGURE 1.7 –

Mis à jour d'un document spécifique

Avant la modification

NewDatabase

Second\_Collection\_test

admin

config

gi2

library

local

localhost:27018

localhost:27019

localhost:27021,localhost:27022,localh...

ADD DATA

EXPORT DATA

UPDATE

DELETE

\_id: ObjectId('67da2e33d8cdeb0662b05627')

name: "John Doe"

age: 29

email: "johndoe@example.com"

address: Object

\_id: ObjectId('67da2e8ffe43d06e24902972')

name: "Reda kadiiri"

age: 29

email: "waaareda@example.com"

address: Object

```

public void updateDocument(String collectionName, String name, String newEmail) { no usages
    MongoDB database = ConnectionInstance.getConnexion().getDatabase("NewDatabase");
    MongoCollection<Document> collection = database.getCollection(collectionName);

    // Filtrer le document par nom
    Document filter = new Document("name", name);

    // Mettre à jour le champ "email" du document trouvé
    Document update = new Document("$set", new Document("email", newEmail));

    try {
        // Mettre à jour le premier document qui correspond au filtre
        collection.updateOne(filter, update);

        System.out.println("Le document avec le nom " + name + " a été mis à jour avec le nouvel email : " + newEmail);
    } catch (Exception e) {
        System.err.println("Erreur lors de la mise à jour du document : " + e.getMessage());
    }
}

```

Après la modification :

```

Mar 13, 2023 2:41:12 AM com.mongodb.internal.diagnostics.logging.Loggers shouldUseSLF4J
WARNING: SLF4J not found on the classpath. Logging is disabled for the 'org.mongodb.driver' component
Le document avec le nom 'Reda kadiiri' a été mis à jour avec le nouvel email : NewReda@gmail.com

```

```

    _id: ObjectId('67da2e8ffe43d06e24902972')
    name: "Reda kadiri"
    age: 29
    email: "NewReda@gmail.com"
    ▶ address: Object

```

## 1.2 Question 7 :

On va ajouter une methode qui calcule le numero de document dans une collection specifique :

```

public long countDocumentsInCollection(String collectionName) { no usages
    MongoDBDatabase database = ConnectionInstance.getConnexion().getDatabase("NewDatabase");
    MongoCollection<Document> collection = database.getCollection(collectionName);

    long count = collection.countDocuments();
    System.out.println("La collection '" + collectionName + "' contient " + count + " documents.");
    return count;
}

```

WARNING: SLF4J not found on the classpath. Logging is disabled for the 'org.mongodb.driver' component  
 La collection 'Second\_Collection\_test' contient 2 documents.

NewDatabase

Second\_Collection\_test

admin

config

gi2

library

local

localhost:27018

localhost:27019

localhost:27021,localhost:27022,localh...

ADD DATA

EXPORT DATA

UPDATE

DELETE

```

_id: ObjectId('67da2e33d8cdeb0662b05627')
name: "John Doe"
age: 29
email: "johndoe@example.com"
▶ address: Object

```

```

_id: ObjectId('67da2e8ffe43d06e24902972')
name: "Reda kadiri"
age: 29
email: "NewReda@gmail.com"
▶ address: Object

```