Crud avec Spring Boot et MongoDB

Ndongo Tonux Samb
Apr 15, 2019 · 4 min read

Ziar naleen sama gars yi (Juste pour dire Bonjour)

Aujourd'hui nous allons apprendre à créer une application Spring Boot avec MongoDB et à effectuer les opérations de base avec Spring JPA et Spring MVC.

Spring Boot ???

Spring Boot est un framework qui facilite le développement d'applications fondées sur Spring en offrant des outils permettant d'obtenir une application packagée, totalement autonome.

MongoDB ???

MongoDB est un système de gestion de base de données ou SGBD, comme **Mysql** ou **PostgreSql**, mais dont le mécanisme est complètement différent. Fini le temps ou il fallait créer un schéma de tables relationnelles et créer des requêtes Sql complexes. Grâce à MongoDB vous allez pouvoir stocker vos données un peu comme vous le feriez dans un fichier JSON. C'est à dire, une sorte de dictionnaire géant composé de clés et de valeurs. Ces données peuvent ensuite être exploitées par du javascript, directement intégré dans MongoDB, mais peuvent également être exploitées par d'autre langage comme python et bien sur Java :) .
En tant que base de données de type ***NoSQL***, MongoDB stocke les données sous la forme d'un document. Ainsi, il offre plus de flexibilité.

Athiaa bokk... on code!!!

On aime pas trop parler. On va suivre ensemble ce tutoriel pour faire un crud. Je vais essayer d'expliquer en détail chaque partie.

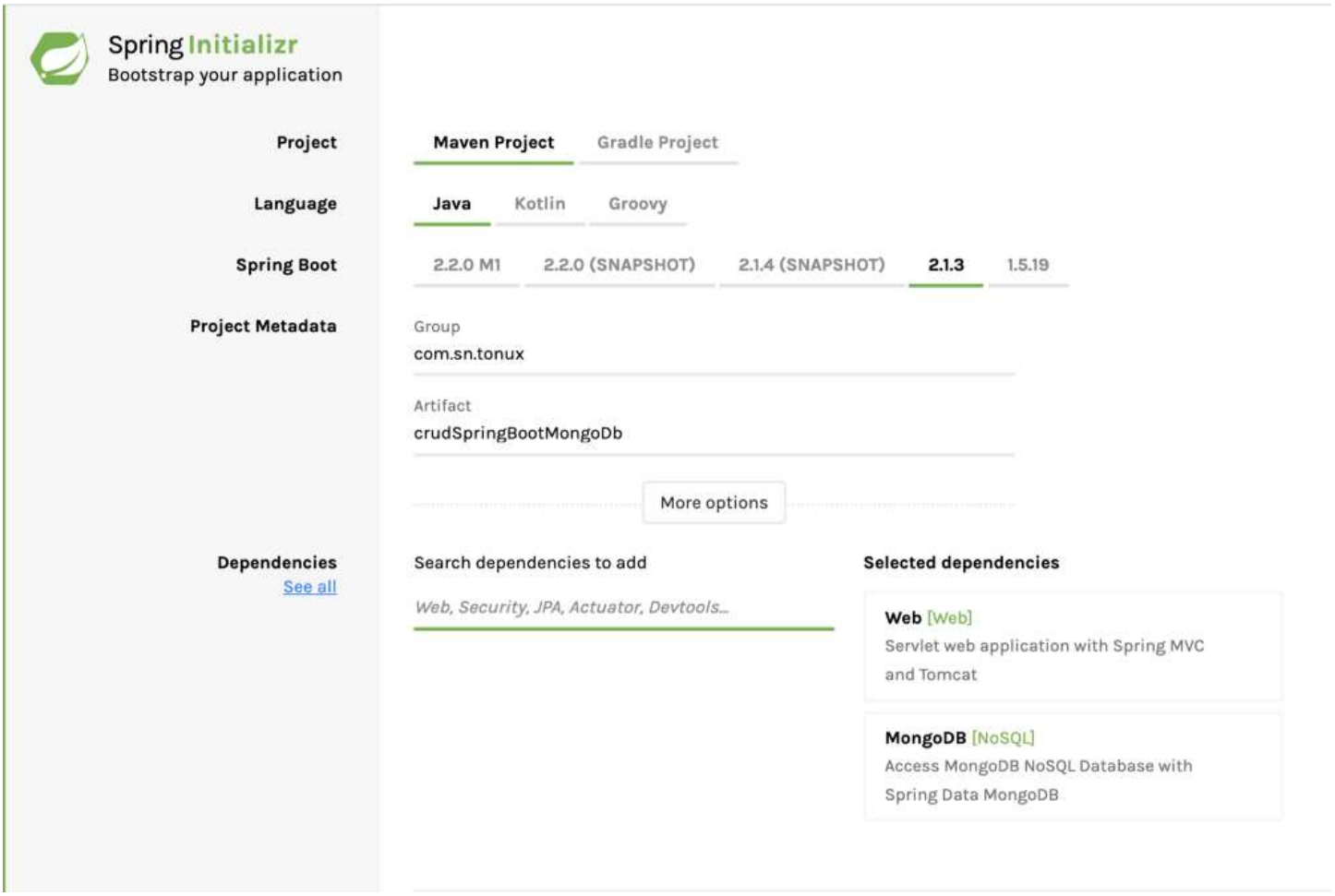
1- pré requis :

Nous avons noté quelques besoins pour réaliser le projet.

- java, JDK 8
- maven
- MongoDB
- (J'ai utilisé intelliJ comme IDE)
- Postman

2- Créer le projet

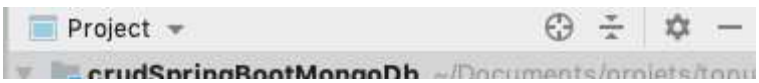
On va utiliser ***https://start.spring.io/*** pour générer notre projet. On a ajouté comme dépendances web et mongoDb.

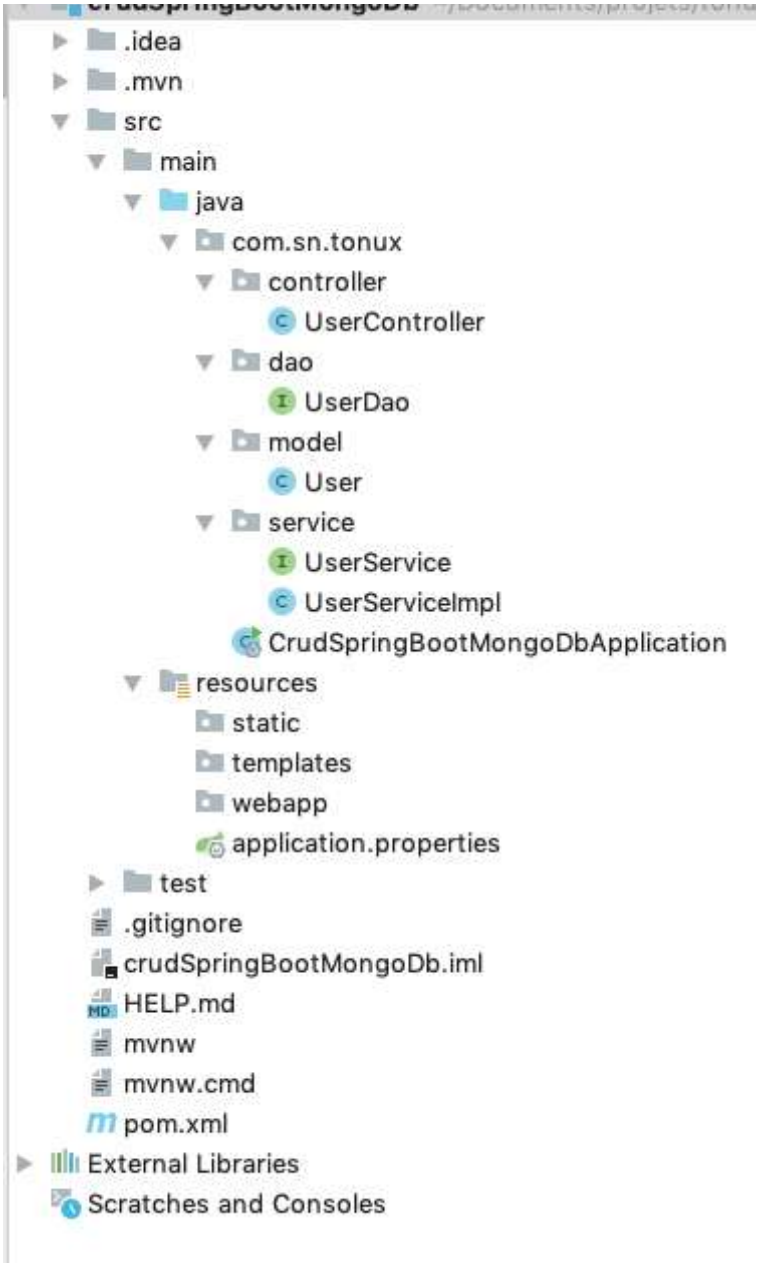


génération du projet

3- Structure du projet

Sous intelliJ voici notre structure du projet. Cela peut changer suivant l'IDE. (Eclipse 😭)





structure du projet

4- Nos Dependences Maven

Voici après génération le contenu de notre fichier ***pom.xml.***

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSc
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/mave
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.1.3.RELEASE</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.sn.tonux</groupId>
    <artifactId>crudSpringBootMongoDb</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>crudSpringBootMongoDb</name>
    <description>Demo project for Spring Boot</description>

    <properties>
        <java.version>1.8</java.version>
```

```xml
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-mongodb</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>

</project>
```

5- Notre fichier Application Properties

Pour les config de la base de donnée, on a utilisé le fichier ***application.properties*** qui se trouve dans resources.

```
# Application configuration.
server.port=8088

# mongodb configuration.
spring.data.mongodb.host=localhost
spring.data.mongodb.port=27017
spring.data.mongodb.database=testDb

# Logging configuration.
logging.level.com.assignment.springboot.mongo=DEBUG
logging.pattern.console= %d{yyyy-MM-dd HH:mm:ss} - %msg%n

```

application.properties

6- Classes Java

- Main class : ***CrudSpringBootMongoDbApplication.java*** : Rappelez-vous toujours que le point d'entrée de l'application Spring Boot est la classe contenant l'annotation ***@SpringBootApplication*** et la méthode main en statique.

```java
package com.sn.tonux;

import ...

@SpringBootApplication
public class CrudSpringBootMongoDbApplication {
```

```
    public static void main(String[] args) { SpringApplication.run(CrudSpringBootMongoDbApplication.class, args); }

}
```

CrudSpringBootMongoDbApplication.java

- Model class : ***User.java***

```
package com.sn.tonux.model;

import org.springframework.data.annotation.Id;
import org.springframework.data.mongodb.core.mapping.Document;

@Document(collection= "user")

public class User {

    @Id
    private int id;
    private String nom;
    private String prenom;
    private String age;


    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getNom() {
        return nom;
    }

    public void setNom(String nom) {
        this.nom = nom;
    }

    public String getPrenom() {
        return prenom;
    }

    public void setPrenom(String prenom) {
        this.prenom = prenom;
    }

    public String getAge() {
        return age;
    }

    public void setAge(String age) {
        this.age = age;
    }


    @Override
    public String toString() {
        return "User (id=" + id + ", nom=" + nom + ", prenom=" + prenom +", age=" + age + ")";
    }
}
```

User.java

- Data-Access-Object interface : ***UserDao.java*** interface qui étend **MongoRepository** pour gérer automatiquement nos requêtes crud.

```java
package com.sn.tonux.dao;

import com.sn.tonux.model.User;
import org.springframework.data.mongodb.repository.MongoRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface UserDao extends MongoRepository<User, Integer> {


}

```

userDao.java

- Service class : ***UserServiceImpl.java*** : classe de service où nous appellerons les méthodes de l'interface Dao pour gérer les opérations SQL. Elle implémente notre interface ***UserService.java***

```java
package com.sn.tonux.service;

import com.sn.tonux.model.User;

import java.util.Collection;
import java.util.List;
import java.util.Optional;

public interface UserService {



    public void createUser(List<User> emp);


    public Collection<User> getAllUsers();


    public Optional<User> findUserById(int id);


    public void deleteUserById(int id);


    public void updateUser(User user);


    public void deleteAllUsers();
}

```

UserService.java

```java
package com.sn.tonux.service;

import com.sn.tonux.dao.UserDao;
import com.sn.tonux.model.User;
import org.springframework.beans.factory.annotation.Autowired;

import java.util.Collection;
import java.util.List;
import java.util.Optional;

```

```java
public class UserServiceImpl implements UserService {

    @Autowired
    UserDao dao;


    @Override
    public void createUser(List<User> user) {
        dao.saveAll(user);
    }


    @Override
    public Collection<User> getAllUsers() {
        return dao.findAll();
    }


    @Override
    public Optional<User> findUserById(int id) {
        return dao.findById(id);
    }


    @Override
    public void deleteUserById(int id) {
        dao.deleteById(id);
    }


    @Override
    public void updateUser(User emp) {
        dao.save(emp);
    }


    @Override
    public void deleteAllUsers() {
        dao.deleteAll();
    }
}
```

UserServiceImpl.java

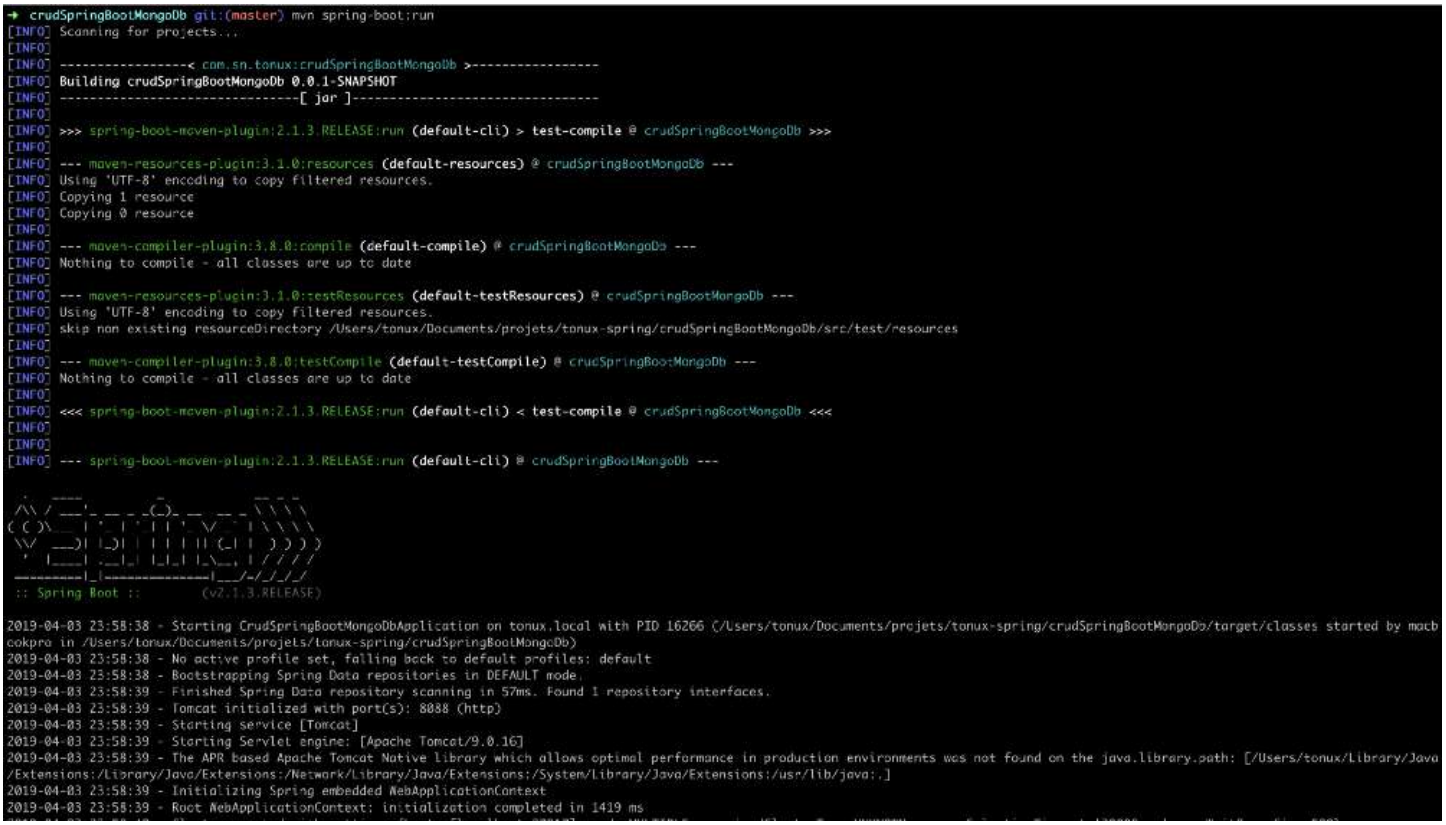
- Controller class : UserController.java : permet de faire le traitement des demandes. La classe est annotée avec **@RestController** où chaque méthode retourne un objet de domaine sous la forme d'une réponse **json** au lieu d'une vue.

```java
package com.sn.tonux.controller;


import java.util.Collection;
import java.util.List;
import java.util.Optional;

import com.sn.tonux.model.User;
import com.sn.tonux.service.UserService;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.*;


@RestController
```

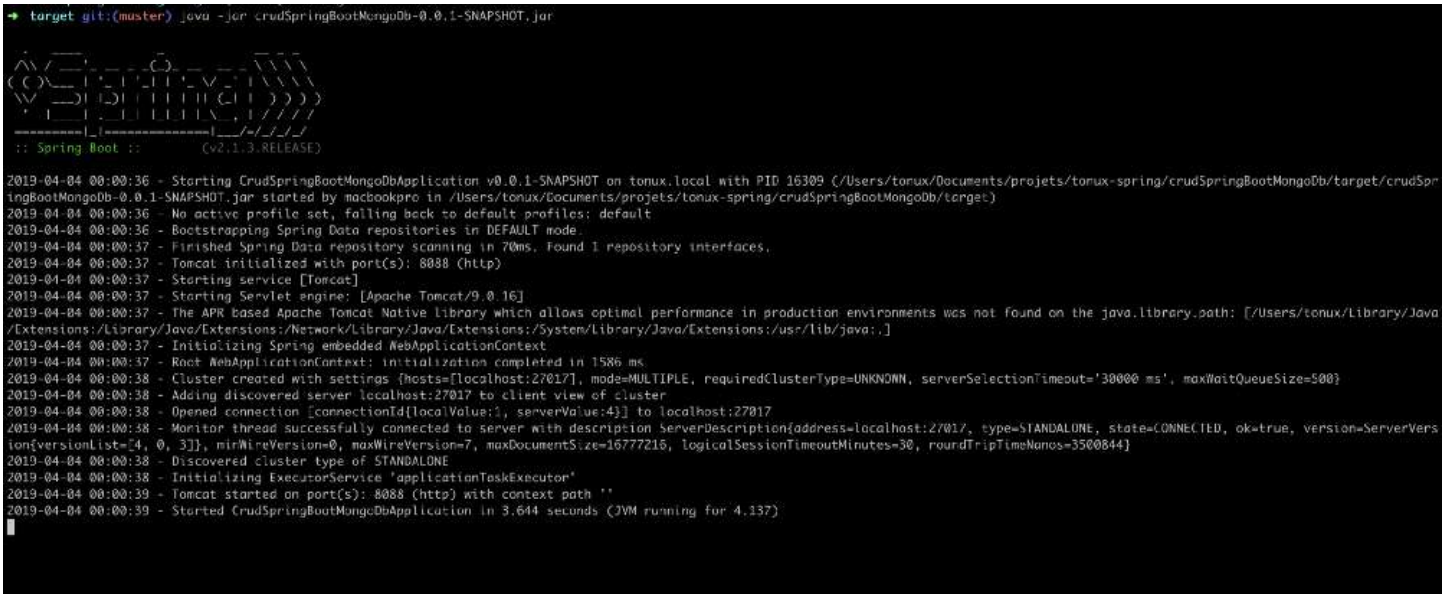
```java
@RequestMapping("/api/v1/users")
public class UserController {



    @Autowired
    @Qualifier(value = "userService")
    UserService serv;

    private final Logger logger = LoggerFactory.getLogger(this.getClass());


    /**
     * Method to fetch all users from the db.
     * @return
     */
    @GetMapping
    @ResponseStatus(HttpStatus.OK)
    public Collection<User> getAll() {
        System.out.println("--------> : getAllUsers");
        logger.debug("Getting all users.");
        return serv.getAllUsers();
    }

    /**
     * Method to fetch user by id.
     * @param id
     * @return
     */
    @GetMapping("/{id}")
    public Optional<User> getById(@PathVariable(value= "user-id") int id) {
        logger.debug("Getting users with user-id= {}.", id);
        return serv.findUserById(id);
    }

    /**
     * Method to update user by id.
     * @param id
     * @param user
     * @return
     */
    @PutMapping("/{id}")
    @ResponseStatus(HttpStatus.OK)
    public String update(@PathVariable(value= "id") int id, @RequestBody User user) {
        logger.debug("Updating user with user-id= {}.", id);
        user.setId(id);
        serv.updateUser(user);
        return "user record for user-id= " + id + " updated.";
    }

    /**
     * Method to delete user by id.
     * @param id
     * @return
     */
    @DeleteMapping("/{id}")
    @ResponseStatus(HttpStatus.OK)
    public String delete(@PathVariable(value= "id") int id) {
        logger.debug("Deleting user with user-id= {}.", id);
        serv.deleteUserById(id);
        return "user record for user-id= " + id + " deleted.";
    }

    /**
     * Method to delete all users from the db.
     * @return
     */
    @DeleteMapping(value= "/deleteall")
    public String deleteAll() {
        logger.debug("Deleting all users.");
        serv.deleteAllUsers();
        return "All users records deleted.";
    }

}
```

Démostration (thiaw sa khirr)

run : *mvn spring-boot:run*



ou bien aller dans target et faire . ***java -jar crudSpringBootMongoDb-0.0.1-SNAPSHOT.jar***



// Créer new user.
http://localhost:8088/api/mongo/users/create

// Get all users.
http://localhost:8088/api/mongo/users

// Find user by id.
http://localhost:8088/api/mongo/users/1

// Update user by id.
http://localhost:8088/api/mongo/users/5

// Delete user by id.
http://localhost:8088/api/mongo/users/1

// Delete all uers.
http://localhost:8088/api/mongo/users/deleteall

Repository du projet

https://github.com/tonux/crud-springboot-mongoDB

```
http://localhost:8102/api/mongo/emp/delete/1

// Delete all employees.

http://localhost:8102/api/mongo/emp/deleteall
```

Spring Spring Boot Mongodb