Spring Boot avec MongoDB

MongoDB

MongoDB est un système de gestion de base de données orienté documents, répartissable sur un nombre quelconque d'ordinateurs et ne nécessitant pas de schéma prédéfini des données. Il est écrit en C++

Nous allons prendre l'application de ToDoList pour l'utilisée avec la base de données.

Nous allons d'abord commencer à configurer le fichier pom.xml

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

Configurons le fichier application.properties

```
# Configuration de MongoDB
spring.data.mongodb.uri=mongodb://localhost:27017/test1
spring.main.allow-bean-definition-overriding=true
```

Le model

```java
package crud.example.demo.model;

import org.springframework.data.mongodb.core.mapping.Document;
import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.ResponseStatus;

@Document(collection = "todolist")
public class ToDo {
    @org.springframework.data.annotation.Id
    private String id;  // MongoDB génère automatiquement un id de type String

    private String titre;
    private String description;
    private boolean status;

    @ResponseStatus(HttpStatus.NOT_FOUND)
    public class ResourceNotFoundException extends RuntimeException {
        public ResourceNotFoundException(String message) {
            super(message);
        }
    }

    public ToDo() {}

    public ToDo(String titre, String description) {
        this.titre = titre;
        this.description = description;
        this.status = false;
    }
```

```java
    // Getters et Setters
    public String getId() { return id; }
    public String getTitre() { return titre; }
    public void setTitre(String titre) { this.titre = titre; }
    public String getDescription() { return description; }
    public void setDescription(String description) { this.description = description; }
    public boolean isStatus() { return status; }
    public void setStatus(boolean status) { this.status = status; }
}
```

Le controller

```java
package crud.example.demo.controller;

import crud.example.demo.model.ToDo;
import crud.example.demo.service.ToDoService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Pageable;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.annotation.*;

//import java.util.HashMap;
import java.util.List;
//import java.util.Optional;

@RestController
@RequestMapping("/lister")
@CrossOrigin(origins = "*")
public class ToDoController {
    @Autowired
    private ToDoService toDoService;

    @ResponseStatus(HttpStatus.NOT_FOUND)
    public class ResourceNotFoundException extends RuntimeException {
        public ResourceNotFoundException(String message) {
            super(message);
        }
    }
```

```java
    @GetMapping
    public List<ToDo> getAllTasks(Pageable pageable) {
        return (List<ToDo>) toDoService.getAllToDos(pageable);
    }

    @GetMapping("/{id}")
    public ResponseEntity getTaskById(@PathVariable Long id) {
        return ResponseEntity.ok(toDoService.getToDoById(id));
    }

    @PostMapping
    public ResponseEntity<ToDo> createTask(@RequestBody @Validated ToDo toDo) {
        ToDo createdToDo = toDoService.createToDo(toDo);
        return new ResponseEntity<>(createdToDo, HttpStatus.CREATED);
    }

    @PutMapping("/{id}")
    public ResponseEntity<ToDo> updateTask(@PathVariable Long id, @RequestBody ToDo toDo) {
        ToDo updatedToDo = toDoService.updateToDo(id, toDo);
        return ResponseEntity.ok(updatedToDo);
    }

    @PatchMapping("/{id}")
    public ResponseEntity<ToDo> updateTache(@PathVariable Long id, @Validated ToDo toDo){
        ToDo updateTache = toDoService.updateToDo(id, toDo);
        return ResponseEntity.ok(updateTache);
    }
```

```java
    @DeleteMapping("/{id}")
    public void deleteTask(@PathVariable Long id) {
        toDoService.deleteToDo(id);
    }

}
```

Le service

```java
package crud.example.demo.service;

import crud.example.demo.model.ToDo;
import crud.example.demo.repository.ToDoRepository;
import org.springframework.stereotype.Service;
import org.springframework.web.bind.annotation.ResponseStatus;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.dao.DuplicateKeyException;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;

import java.util.Optional;

@Service
public class ToDoService {
    @Autowired
    private ToDoRepository toDoRepository;

    @ResponseStatus(HttpStatus.NOT_FOUND)
    public class ResourceNotFoundException extends RuntimeException {
        public ResourceNotFoundException(String message) {
            super(message);
        }
    }
```

```java
  // Récupérer toutes les tâches avec pagination
  public Page<ToDo> getAllToDos(Pageable pageable) {
      return toDoRepository.findAll(pageable);
  }

  // Récupérer une tâche par ID
  public ResponseEntity<?> getToDoById(Long id) {
      Optional<ToDo> toDo = toDoRepository.findById(id);
      return toDo.map(ResponseEntity::ok)
                 .orElseThrow(() -> new ResourceNotFoundException("Tâche non trouvée avec l'ID : " + id));
  }

  // Créer une nouvelle tâche
  public ToDo createToDo(ToDo toDo) {
      if (toDoRepository.existsByTitre(toDo.getTitre())) {
          throw new DuplicateKeyException(msg:"Une tâche avec ce titre existe déjà");
      }
      return toDoRepository.save(toDo);
  }
```

```java
    // Mettre à jour une tâche
    public ToDo updateToDo(Long id, ToDo updateToDo) {
        return toDoRepository.findById(id).map(toDo -> {
            toDo.setTitre(updateToDo.getTitre());
            toDo.setDescription(updateToDo.getDescription()); // Correction
            toDo.setStatus(updateToDo.isStatus());
            return toDoRepository.save(toDo);
        }).orElseThrow(() -> new ResourceNotFoundException("Tâche non trouvée avec l'ID : " + id));
    }

    // Supprimer une tâche
    public void deleteToDo(Long id) {
        if (!toDoRepository.existsById(id)) {
            throw new ResourceNotFoundException("Tâche non trouvée avec l'ID : " + id);
        }
        toDoRepository.deleteById(id);
    }
}
```

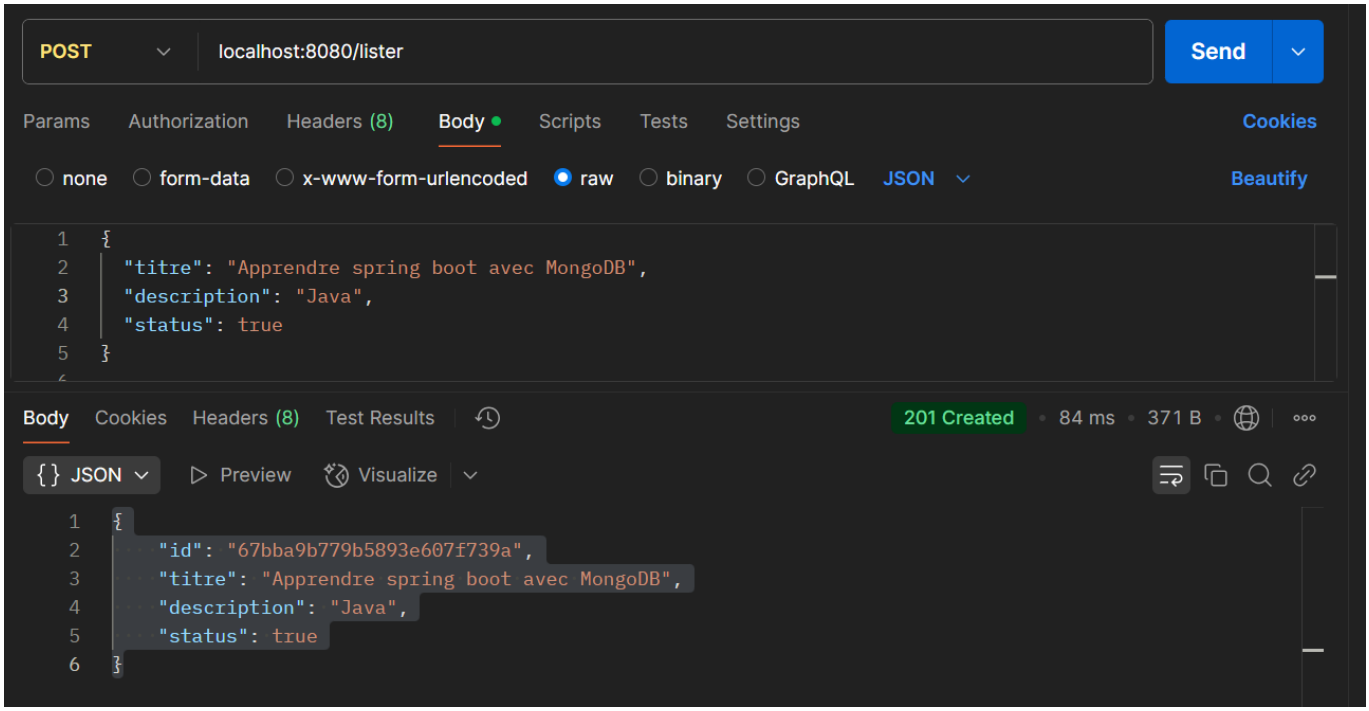
Le repository

```java
package crud.example.demo.repository;

import crud.example.demo.model.ToDo;

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

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.data.mongodb.repository.MongoRepository;

/*public interface ToDoRepository extends JpaRepository<ToDo, Long>{
    boolean existsByTitre(String titre);
}*/

public interface ToDoRepository extends MongoRepository<ToDo, String> {
    boolean existsByTitre(String titre);
    Optional<ToDo> findById(Long id);
    boolean existsById(Long id);
    void deleteById(Long id);
}
```

On démarre l'application mvn spring-boot:run

```
PS C:\Users\PC\Downloads\demo\demo> mvn spring-boot:run
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------< crud.example:demo >-------------------------
[INFO] Building demo 0.0.1-SNAPSHOT
[INFO]   from pom.xml
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] >>> spring-boot:3.4.3:run (default-cli) > test-compile @ demo >>>
[INFO]
[INFO] --- resources:3.3.1:resources (default-resources) @ demo ---
```

La méthode POST



Nous allons afficher les données sur le serveur MongoDB