

## Activité pratique : Gestion des patients

**Créé par : Mohamed Ait Lahcen**

# L'énoncé de l'activité :

## Activité Pratique Spring MVC Thymeleaf

Mohamed YOUSSEFI • 22 mars

100 points

Voir la vidéo : Créer une application Web JEE basée sur Spring MVC, Thymeleaf et Spring Data JPA qui permet de gérer les patients. L'application doit permettre les fonctionnalités suivantes :

- Afficher les patients
- Faire la pagination
- Chercher les patients
- Supprimer un patient
- Faire des améliorations supplémentaires

### 1. Le fichier configuration :

```
#spring.datasource.url=jdbc:h2:mem:patients-mvc
server.port=8086
#spring.h2.console.enabled=true
spring.datasource.url = jdbc:mysql://localhost:3306/patients-mvc
spring.datasource.username = root
spring.datasource.password =
spring.datasource.driverClassName = com.mysql.jdbc.Driver
spring.jpa.show-sql = true
spring.jpa.hibernate.ddl-auto = update
spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL5Dialect
spring.thymeleaf.cache=false
```

## 2. L'entité Patient :

8 usages

@Entity

@Data @NoArgsConstructor @AllArgsConstructor

public class Patient {

@Id @GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String nom;

@Temporal(TemporalType.DATE)

private Date date\_naissance;

private boolean malade;

private int score;

}

- La table Patient dans la base de données :

#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra	Action
<input type="checkbox"/> 1	id	bigint(20)			Non	Aucun(e)		AUTO_INCREMENT	Modifier  Supprimer ▼ Plus
<input type="checkbox"/> 2	date_naissance	date			Oui	NULL			Modifier  Supprimer ▼ Plus
<input type="checkbox"/> 3	malade	bit(1)			Non	Aucun(e)			Modifier  Supprimer ▼ Plus
<input type="checkbox"/> 4	nom	varchar(255) utf8mb4_general_ci			Oui	NULL			Modifier  Supprimer ▼ Plus
<input type="checkbox"/> 5	score	int(11)			Non	Aucun(e)			Modifier  Supprimer ▼ Plus

## 4. La classe PatientRepository :

```
4 usages
public interface PatientRepository extends JpaRepository<Patient,Long> {
    Patient findByNom(String name);
1 usage
    Page<Patient> findByNomContains(String kw,Pageable pageable);
}
```

## 5. PatientController :

cette classe représente le contrôleur de l'application qui s'occupe de répondre aux requêtes du client en lui transmettant les vues.

```
@Controller
@ControllerAdvice
public class PatientController {
    3 usages
    private PatientRepository patientRepository;
```

## ➤ Les fonctionnalités de l'application :

### 1. L'affichage des patients :

voici la fonction du contrôleur qui permet de récupérer la liste des patients :

```
@GetMapping("/index")  
public String patients(Model model,@RequestParam(name = "page",defaultValue = "0") int page,@RequestParam(name = "size",default  
    Page<Patient> pagePatients = patientRepository.findByNomContains(keyWord,PageRequest.of(page,size));  
    model.addAttribute( attributeName: "listPatients",pagePatients.getContent());
```






### • La vue patients.html :

voici une partie de code du fichier html qui permet d'afficher la liste des patients:

```
<!DOCTYPE html>  
<html xmlns:th="http://www.thymeleaf.org">  
<head>  
    <meta charset="utf-8"/>  
    <title>Insert title here</title>  
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.3.0/font/bootstrap-icons.css">  
    <link rel="stylesheet" type="text/css" href="webjars/bootstrap/5.1.3/css/bootstrap.min.css"/>  
    <style>  
        .inp{  
            height: 40px;  
            border-radius: 5px;  
        }  
    </style>
```

## La liste des patients



ID	Nom	Date de Naissance	Malade	Score	Delete
2	patient2	2022-03-29	false	160	
3	patient3	2022-03-29	true	220	
4	patient4	2022-03-29	true	320	
5	patient1	2022-04-03	false	120	
6	patient2	2022-04-03	false	160	

0

1

2

3

4

5

6

7

8

9

▶

## 2. La pagination :

Dans l'interface graphique(vue) on affiche dix pages et deux bouton un pour passer à la page suivante et un autre bouton pour la page précédente.

```

int nbrPage;
if(pagePatients.getTotalPages()>10)nbrPage=10;
else nbrPage=pagePatients.getTotalPages();
int pages[]=new int[nbrPage];
if(page<7){
    for(int i=0;i<nbrPage;i++){
        pages[i]=i;
    }
}
else if(page>=7 && page<pagePatients.getTotalPages()-5){
    for(int j=0;j<nbrPage;j++){
        pages[j]=page-5+j;
    }
}
}

```

```

else if(page>=pagePatients.getTotalPages()-5){
    for(int k=0;k<nbrPage;k++){
        pages[k]=pagePatients.getTotalPages()-10+k;
    }
}

model.addAttribute(attributeName: "pages", pages);
model.addAttribute(attributeName: "currentPage", page);

```

### 3. La suppression d'un patient :

Voici la méthode qui permet de supprimer un patient :

```

@GetMapping("/delete")
public String delete(Long id,int page,String keyWord){
    patientRepository.deleteById(id);
    return "redirect:/index?page="+page+"&keyWord="+keyWord;
}

```

#### 4. La recherche des patients :

voici la partie de code dans la fonction du contrôleur index qui permet de récupérer les patients par le mot clé :

```
public String patients(Model model, @RequestParam(name = "page", defaultValue = "0") int page, @RequestParam(name = "keyword") String keyword) {  
    Page<Patient> pagePatients = patientRepository.findByNomContains(keyword, PageRequest.of(page, size));  
    return "patients";  
}
```

#### 5. Les améliorations supplémentaires :

- **L'amélioration de la pagination :**

pour faire la pagination de cette application j'ai suivi les normes de la pagination en fait à chaque fois j'affiche dix pages et deux boutons une pour basculer vers la page suivante et une autre pour basculer vers la page précédente :



- **L'utilisation des icons :**

Dans cette application j'ai utilisé les icons de librairie bootstrap :

```
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.3.0/font/bootstrap-icons.css">
```