

DEPARTEMENT MATHÉMATIQUES ET INFORMATIQUE

Compte rendu 5

JEE

Filière :
« Génie du Logiciel et des Systèmes Informatiques Distribués »
GLSID

Le 25/03/2022

Préparé par : Ahlam CHAKRANE

Année Universitaire : 2020-2021

Introduction

Ce TP a pour objectif de mettre en évidence l'utilisation des JSP et l'architecture Spring MVC coté client.

Dans ce TP, nous allons faire la gestion des patients, et nous allons traiter diverses actions notamment : l'ajout, la modification, la suppression, la recherche et la pagination.

Pour mettre en place notre application, nous devons créer tout d'abord la classe Patient :

```
Patient.java × PatientRepository.java Seance6Application.java
1 package ma.enset.Seance6.entities;
2
3 import java.util.Date;
4
5 import javax.persistence.Entity;
6 import javax.persistence.GeneratedValue;
7 import javax.persistence.GenerationType;
8 import javax.persistence.Id;
9 import javax.persistence.Temporal;
10 import javax.persistence.TemporalType;
11
12 import lombok.AllArgsConstructor;
13 import lombok.Data;
14 import lombok.NoArgsConstructor;
15 @Entity
16 @Data @AllArgsConstructor @NoArgsConstructor
17 public class Patient {
18     @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
19     private Long id;
20     private String nom;
21     @Temporal(TemporalType.DATE)
22     private Date dateNaissance;
23     private boolean malade;
24     private int score;
25 }
26
```

Les annotations `@NoArgsConstructor` et `@AllArgsConstructor` permet de générer automatiquement des constructeurs sans et avec paramètres.

L'annotation `@Id` permet de déclarer de l'attribue est une clé primaire de la table de la base de données.

Ensuite nous avons créé un Repository qui hérite de `JpaRepository` pour générer automatiquement les méthodes classiques qui permettent de gérer les entités JPA.

```
Patient.java × PatientRepository.java × Seance6Application.java application.properties
1 package ma.enset.Seance6.repositories;
2
3 import org.springframework.data.jpa.repository.JpaRepository;
4
5 import ma.enset.Seance6.entities.Patient;
6
7 public interface PatientRepository extends JpaRepository<Patient, Long>{
8
9 }
10
```

Dans la fonction main, nous avons ajouté des patients à stocker dans la base de données.

```
Patient.java PatientRepository.java Seance6Application.java application.properties
1 package ma.enset.Seance6;
2
3 import java.util.Date;
12
13 @SpringBootApplication
14 public class Seance6Application {
15
16     public static void main(String[] args) {
17         SpringApplication.run(Seance6Application.class, args);
18     }
19
20     @Bean
21     CommandLineRunner commandeLineRunner(PatientRepository patientRepository) {
22         return args -> {
23             patientRepository.save(new Patient(null, "ahlam", new Date(), true, 30));
24             patientRepository.save(new Patient(null, "aicha", new Date(), false, 10));
25             patientRepository.save(new Patient(null, "taoufik", new Date(), false, 20));
26             patientRepository.save(new Patient(null, "anas", new Date(), true, 100));
27             patientRepository.findAll().forEach(p -> {
28                 System.out.println(p.getNom());
29             });
30         };
31     }
32 }
```

Dans le fichier application.properties, nous avons déclaré la base de données à utiliser, et le port

```
Patient.java PatientRepository.java Seance6Application.java application.properties
1 spring.datasource.url=jdbc:h2:mem:patients_db
2 spring.h2.console.enabled=true
3 server.port=8082
4
```

Après avoir tourné l'application, les données ont été stockées dans notre base de données qui porte le nom « patients_db » :

```
2022-03-24 09:03:19.813 INFO 4716 --- [main]
ahlam
aicha
taoufik
anas
2022-03-24 09:04:27.824 INFO 4716 --- [nio-8082-exec-1]
2022-03-24 09:04:27.824 INFO 4716 --- [nio-8082-exec-1]
```

Alors si on accède à la base de données H2 :

English ▼ Preferences Tools Help

Login

Saved Settings: Generic H2 (Embedded) ▼

Setting Name: Generic H2 (Embedded) Save Remove

Driver Class: org.h2.Driver

JDBC URL: jdbc:h2:mem:patients_db

User Name: sa

Password:

Connect Test Connection

On trouve les enregistrements suivants :

Run Run Selected Auto complete Clear SQL statement:

SELECT * FROM PATIENT|

SELECT * FROM PATIENT;

ID	DATE_NAISSANCE	MALADE	NOM	SCORE
1	2022-03-24	TRUE	ahlam	30
2	2022-03-24	FALSE	aicha	10
3	2022-03-24	FALSE	taoufik	20
4	2022-03-24	TRUE	anas	100

(4 rows, 12 ms)

Edit

Pour basculer vers MySQL, il faut changer les données du fichier application.properties :

```

Patient.java  Seance6Appli...  patients.html  application....  PatientCont...  addPat
1 #spring.datasource.url=jdbc:h2:mem:patients_db
2 #spring.h2.console.enabled=true
3 #server.port=8082
4 spring.datasource.url=jdbc:mysql://localhost:3306/patients_db?createDatabaseIfNotExist=true
5 spring.datasource.username=root
6 spring.datasource.password=
7 server.port=8082
8 spring.jpa.hibernate.ddl-auto = update
9 spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MariaDBDialect
10 spring.jpa.show = true
11 spring.thymeleaf.cache= false

```

Pour faire la gestion des routes, nous avons créé une classe controleur pour faire le traitement nécessaire lors de l'appel des routes :

La première route :

```

@Controller
@AllArgsConstructor
public class PatientController {
    private PatientRepository patientRepository;
    @GetMapping(path = "/index")
    public String patients(Model model,
        @RequestParam(name= "page", defaultValue = "0") int page,
        @RequestParam(name= "size", defaultValue = "8") int size,
        @RequestParam(name= "keyword", defaultValue = "") String keyword) {

        Page<Patient> pagePatients = patientRepository.findByNomContains(keyword,PageRequest.of(page, size));
        model.addAttribute("ListPatients",pagePatients.getContent());
        model.addAttribute("pages", new int[pagePatients.getTotalPages()]);
        model.addAttribute("currentPage",page);
        model.addAttribute("keyword",keyword);
        return "patients";
    }
    @GetMapping("/delete")

```

Lorsqu'on tape la route /index, on se dirige dans la page patients.html que dans laquelle on affiche 8 patients dans chaque page, puisqu'on utilise la pagination.

Pour se faire, on récupère la liste des patients, et on la stocke dans une variable appelée ListPatients, qu'on la met dans la variable model de type Model.

On passe également le nombre de page pour faire l'affichage des patients selon ce nombre, la page courante, et le mot recherché s'il existe dans la barre de recherche pour ne le pas perdre si l'utilisateur passe d'une page à une autre.

La deuxième route :

```

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

```

Lorsque l'utilisateur click sur le bouton de suppression, il sera dirigé vers cette fonction, que dans la quel on supprime le patient qui a l'id correspondant avec celui du patient sélectionné, et finalement on redirige l'utilisateur vers la page index, tout en gardant la page courante et le mot recherché s'il existe.

La troisième route :

```
@GetMapping("/add")
public String add() {
    return "addPatient";
}
```

Lorsque le client click sur le bouton d'ajout, il sera envoyé vers la vue addPatient.html

La quatrième route :

```
@PostMapping("/save")
public String save(Model model, Patient patient) {
    patientRepository.save(patient);
    return "redirect:/index";
}

@GetMapping("/update")
```

Lorsque l'utilisateur click sur le bouton save pour enregistrer un nouvel utilisateur, il sera dirigé vers cette route pour sauvegarder le patient ajouté, puis le redirigé vers la vue index.html

La cinquième route :

```
@GetMapping("/update")
public String update(Model model, Long id) {
    Patient patient=patientRepository.findById(id).get();
    model.addAttribute("patient",patient);
    return "updatePatient";
}
```

Si l'utilisateur souhaite effectuer des modifications sur un patient, il va être dirigé vers la route update. Un patient qui portera le même id que celui sélectionné sera sélectionné de la base de données, puis stocker dans un objet patient, qu'on stocke dans la variable model de type model. Tous cela pour récupérer ce patient dans la vue updatePatient.html, pour afficher les informations correspondantes à ce patient.

La sixième route :

```
@GetMapping("/")
public String home() {
    return "redirect:/index";
}

/**/
```

Si l'utilisateur ne précise pas de route il sera alors dirigé vers la vue index.html

Alors pour afficher une liste de patients voici la vue patients.html :

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
  <meta charset="UTF-8">
  <title>title</title>
  <link rel="stylesheet" href="/webjars/bootstrap/5.1.3/css/bootstrap.min.css">
  <link rel="stylesheet" href="/webjars/bootstrap-icons/1.8.1/font/bootstrap-icons.css">
</head>
<body>
<div class="container mt-4">
  <div class="card">
    <div class="card-header">
      Liste des patients
      <a class="btn btn-success" th:href="@{add}">
        <i class="icon-eye">+</i>
      </a>
    </div>
    <div class="card-body">
      <form method="get" th:action="@{index}">
        <label>Key word</label>
        <input type="text" name="keyword" th:value="${keyword}">
        <button type="submit" class="btn btn-primary">Search</button>
      </form>
      <table class="table">
        <thead>
          <tr>
            <th>ID</th>
            <th>Nom</th>
            <th>Date</th>
            <th>Malade</th>
            <th>Score</th>
            <th>Action</th>
          </tr>
        </thead>
        <tbody>
          <tr th:each="p:${ListPatients}">
            <td th:text="{p.id}"></td>
            <td th:text="{p.nom}"></td>
            <td th:text="{p.dateNaissance}"></td>
            <td th:text="{p.malade}"></td>
            <td th:text="{p.score}"></td>
            <td>
              <a onclick="return confirm('Etes vous sure ?')" class="btn btn-danger" th:href="@{delete(id=${p.id}, k
                <i class="bi bi-trash"></i>
              </a>
              <a class="btn btn-warning" th:href="@{update(id=${p.id})}">
                <i class="bi bi-pen">update</i>
              </a>
            </td>
          </tr>
        </tbody>
      </table>
      <ul class="nav nav-pills">
        <li th:each="page,status:${pages}">
          <a th:class="${status.index==currentPage?'btn btn-info ms-1':'btn btn-outline-info ms-1'}"
            th:text="{status.index}"
            th:href="@{index(page=${status.index}, keyword=${keyword})}">
          </a>
        </li>
      </ul>
    </div>
  </div>
</div>

</body>
</html>
```

```
, keyword=${keyword}, page=${currentPage}}">
```

Pour ajouter un patient voici la vue addPatient.html :

```
1<!DOCTYPE html>
2<html lang="en" xmlns:th="http://www.thymeleaf.org">
3<head>
4  <meta charset="UTF-8">
5  <title>New patient</title>
6  <link rel="stylesheet" href="/webjars/bootstrap/5.1.3/css/bootstrap.min.css">
7</head>
8<body>
9<center>
10  <div class="col-md-6">
11    <div class="panel panel-default">
12      <div class="panel-heading">New Patient</div>
13      <div class="panel-body">
14        <form th:action="@{save}" method="post">
15          <div class="form-group">
16            <label class="control-label">Nom:</label>
17            <input class="form-control" type="text" name="nom">
18          </div>
19          <div class="form-group">
20            <label class="control-label">Date Naissance:</label>
21            <input class="form-control" type="date" name="dateNaissance" value="12/12/2022" required>
22          </div>
23          <div class="form-check">
24            <input class="form-check-input" type="checkbox" name="malade" id="malade"
25              th:checked="${flag} ? 'checked'">
26            <label class="form-check-label" for="malade">
27              Malade
28            </label>
29          </div>
30          <div class="form-group">
31            <label class="control-label">Score:</label>
32            <input class="form-control" type="number" name="score" value="0">
33          </div>
34          <button type="submit" class="btn btn-success">Save</button>
35        </form>
36      </div>
37    </div>
38  </div>
39</center>
40</body>
```

Et pour modifier un patient voici la vue updatePatient.html

```

Seance6Appli... application... PatientCont... *addPatient... updatePatien... X Patient.java »
5 <title>New patient</title>
6 <link rel="stylesheet" href="/webjars/bootstrap/5.1.3/css/bootstrap.min.css">
7 </head>
8 <body>
9 <center>
10 <div class="col-md-6">
11 <div class="panel panel-default">
12 <div class="panel-heading">Update Patient</div>
13 <div class="panel-body">
14 <form th:action="@{/save}" method="post">
15 <div class="form-group">
16 <label class="control-label">ID:</label>
17 <input class="form-control" type="text" name="id" th:value="${patient.id}" readonly="readonly">
18 </div>
19 <div class="form-group">
20 <label class="control-label">Date Naissance:</label>
21 <input class="form-control" type="date" name="dateNaissance" th:value="${patient.dateNaissance}">
22 </div>
23 <div class="form-group">
24 <label class="control-label">Nom:</label>
25 <input class="form-control" type="text" name="nom" th:value="${patient.nom}">
26 </div>
27 <div class="form-check">
28 <input class="form-check-input" type="checkbox" name="malade" id="malade"
29 th:checked="${flag} ? 'checked'">
30 <label class="form-check-label" for="malade">
31 Malade
32 </label>
33 </div>
34 <div class="form-group">
35 <label class="control-label">Score:</label>
36 <input class="form-control" type="number" name="score" th:value="${patient.score}">
37 </div>
38 <button type="submit" class="btn btn-success">Save</button>
39 </form>
40 </div>
41 </div>
42 </div>
43 </center>
44 </body>

```

Alors
notre

















application a pris la forme suivante :

La liste des patients :

Liste des patients +

Key word

Search

ID	Nom	Date	Malade	Score	Action
1	ahlam	2022-03-25	false	12	 
2	aicha	2022-03-25	false	10	 
3	taoufik	2022-03-25	false	20	 
4	anas	2022-03-25	true	100	 
5	ahlam nouv	2022-03-12	true	12	 
6	ahlam modif	2021-03-16	true	100	 
7	ahlam	2022-03-25	true	30	 
8	aicha	2022-03-25	false	10	 

0

1

2

3

4

5

6

7

8

9

















10

Rechercher un patient :

localhost:8082/index?keyword=ahlam

Liste des patients +

Key word Search





ID	Nom	Date	Malade	Score	Action
1	ahlam	2022-03-25	false	12	 
5	ahlam nouv	2022-03-12	true	12	 
6	ahlam modif	2021-03-16	true	100	 
7	ahlam	2022-03-25	true	30	 
11	ahlam	2022-03-25	true	30	 
15	ahlam	2022-03-25	true	30	 
19	ahlam	2022-03-25	true	30	 
23	ahlam	2022-03-25	true	30	 

0 1 2

Supprimer le patient dont l'id est 1 :

localhost:8082 says
Etes vous sure ?

















OK Cancel

Date	Malade	Score	Action
2022-03-25	false	12	 
2022-03-12	true	12	 

Liste des patients après suppression :

Liste des patients +

Key word Search

ID	Nom	Date	Malade	Score	Action
5	ahlam nouv	2022-03-12	true	12	 
6	ahlam modif	2021-03-16	true	100	 
7	ahlam	2022-03-25	true	30	 
11	ahlam	2022-03-25	true	30	 
15	ahlam	2022-03-25	true	30	 
19	ahlam	2022-03-25	true	30	 
23	ahlam	2022-03-25	true	30	 
27	ahlam	2022-03-25	true	30	 

0
1
2

Ajouter un nouveau patient :


localhost:8082/add

New Patient

Nom:

new test


Date Naissance:

03/23/2022 





☐

Malade

Score:

199 

Save

85	anas	2022-03-25	true	100	 
86	new test	2022-03-23	false	199	 

0
1
2
3
4
5
6
7
8
9
10

Modifier un patient :

localhost:8082/update?id=86

Update Patient

ID:

86

Date Naissance:

03/02/2022



Nom:

new test modif







Malade

Score:

19

Save

86	new test modif	2022-03-02	true	19	 
86	new test modif	2022-03-02	true	19	 

0

1

2

3

4

5

6

7

8

9

10

Conclusion :

Grace à ce TP, nous avons pu comprendre l'architecture Spring MVC, et comment créer des JSP.
Nous avons pu mieux développer nos compétences en programmation JEE, ainsi que nous avons pu améliorer notre connaissance de gérer des routes.