Année : 2022/2023

Compte rendu

Application web JEE, Gestion des patients

Filière :

Méthodes Informatiques Appliquées à la Gestion des Entreprises

Realisée par :Kidiss Leila	Encadré par M. Youssfi Mohammed

Sommaire:

Sujet

L’objectif principal consiste à concevoir et réaliser une application web dynamique avec le Framework Spring qui permet la gestion d’un hôpital. Les données sont stockées dans une base de données MySQL

L’application se compose de trois couches :

La couche DAO qui est basée sur Spring Data, JPA, Hibernate et JDBC.

La couche Métier

La couche Web basée sur MVC coté Serveur en utilisant Thymeleaf.

La sécurité est basée sur Spring Security

Annotations et mots clés

<table>
<tr><td>Injection de dépendance</td><td>Permet d'implémenter le principe de l'inversion de contrôle<table><tr><td>Inversion de contrôle</td><td>=> découpler les dépendances</td></tr></table></td></tr>
<tr><td>JDBC
(Java Data Base Connection)</td><td>API qui permet d’utiliser les bases de données relationnelles</td></tr>
<tr><td>Mapping Objet Relationnel
(ORM)</td><td>géré l’accès aux données</td></tr>
</table>

Object-Relational Mapping	

Hibernate	Un ORM qui implémente la spécification JPA
JPA (Java Persistance API)	Un API qui repose essentiellement sur l'utilisation des annotations
Spring Data	Un module de Spring qui facilite l'utilisation de JPA
Spring Data JPA	Fait l'ORM basé sur JPA
Lombok	Permet de générer les getters et les setters
Spring Web	Spring mvc
H2 Database	SGBD à mémoire (les données sont perdus apres chaque redémarrage)
application.properties	fichier de configuration de l'application

Entities	Package qui contient les classes qui vont etre par la suite des tables dans la bd

@Data	annotation de Lombok => gener les getters et setters et constructeurs
@ Entity	=>annotation essentiel pour une classe pour devenir une entité JPA
@ID	Primary key =>annotation essentiel pour une classe pour devenir une entité JPA
repositories	Package qui contient des interfaces qui hérite de l'interface JpaRepository => permet d'utiliser JPA
@Query	Annotation qui dit a SpringData comment interprète la fonction =>Utilise HQL (Hibernate query language)
@Bean	Dit a Spring d'exécuter celle-ci au démarrage

BindingResult	informations sur les erreurs de validation
@Valid	Hibernate va utiliser les annotations de validations (comme @Min et @NotEmpty) avant d'exécuter les requêtes sql
@configuration	Dit a Spring que cette classe doit être instancier au premier lieu

Dpendances

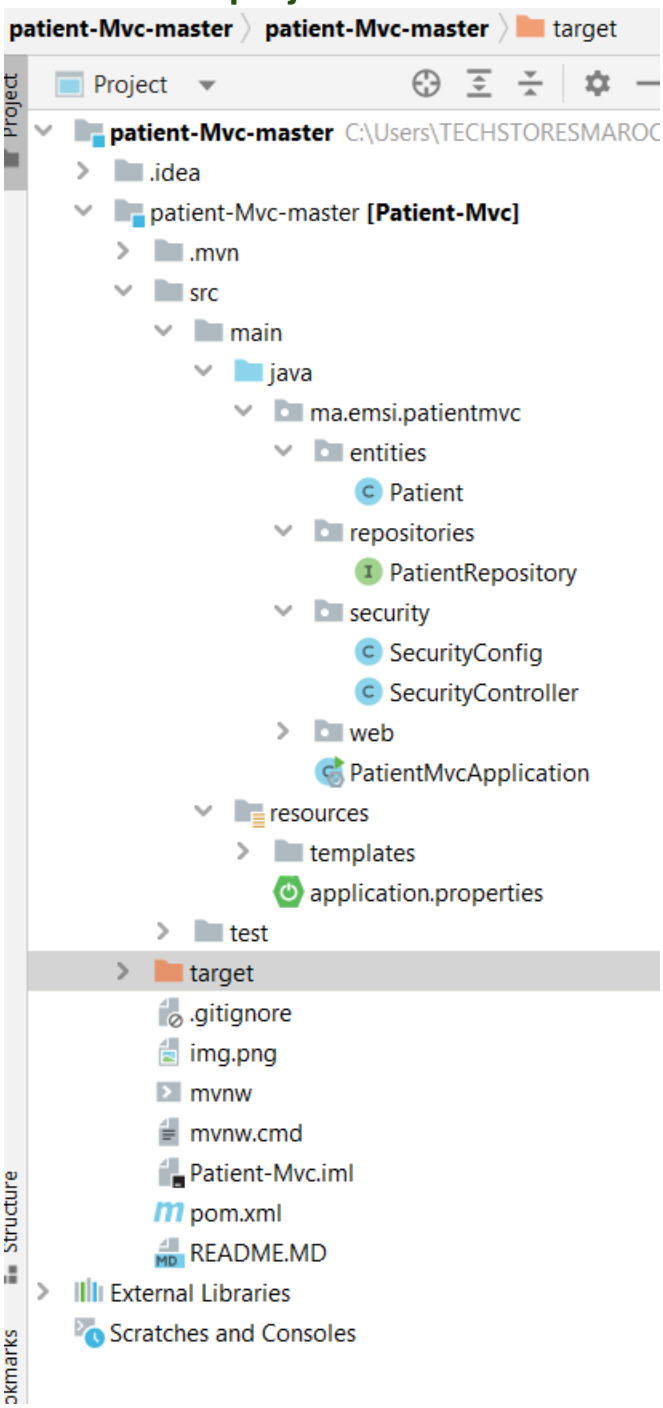
```
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-thymeleaf</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-security</artifactId>
    </dependency>
    <!-- https://mvnrepository.com/artifact/org.webjars/bootstrap -->
    <dependency>
        <groupId>org.webjars</groupId>
        <artifactId>bootstrap</artifactId>
        <version>5.2.3</version>
    </dependency>
    <!-- https://mvnrepository.com/artifact/nz.net.ultraq.thymeleaf/thymeleaf-layout-dialect -
```

```
    <version>5.2.3</version>
</dependency>
<!-- https://mvnrepository.com/artifact/nz.net.ultraq.thymeleaf/thymeleaf-layout-dialect -->
<dependency>
    <groupId>nz.net.ultraq.thymeleaf</groupId>
    <artifactId>thymeleaf-layout-dialect</artifactId>
</dependency>

<dependency>
    <groupId>com.mysql</groupId>
    <artifactId>mysql-connector-j</artifactId>
    <scope>runtime</scope>
</dependency>
    <dependency>-->
        <groupId>com.h2database</groupId>-->
        <artifactId>h2</artifactId>-->
        <scope>runtime</scope>-->
    </dependency>-->
<!-- https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-validation -->
<dependency>
    <groupId>org.springframework.boot</groupId>
```

z

```
    <dependency>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok</artifactId>
        <optional>true</optional>
    </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>
            <configuration>
                <excludes>
                    <exclude>
```

Structure de projet



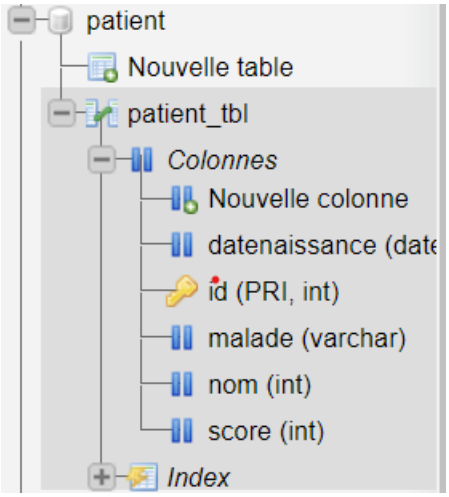
Les Entités

```java
@Entity
@Data
@NoArgsConstructor
@AllArgsConstructor
public class Patient implements Serializable {

    no usages
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    no usages
    @NotEmpty
    @Size(min = 4, max = 40)
    @Column(length = 50)
    private String nom;
    no usages
    @Temporal(TemporalType.DATE)
    @DateTimeFormat(pattern = "yyyy-MM-dd")
    private Date dateNaissance;
    no usages
    private boolean malade;
    no usages
    @DecimalMin("100")
    private int score;
```

Déployer le data source

Le fichier de configuration ‘application.properties’

```
spring.datasource.url=jdbc:mysql://localhost:3306/patient?createDatabaseIfNoExist=true
spring.datasource.username=root
spring.datasource.password=
server.port=8081
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=update
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MariaDBDialect
spring.thymeleaf.cache=false
spring.mvc.format.date=YYYY-MM-dd
```

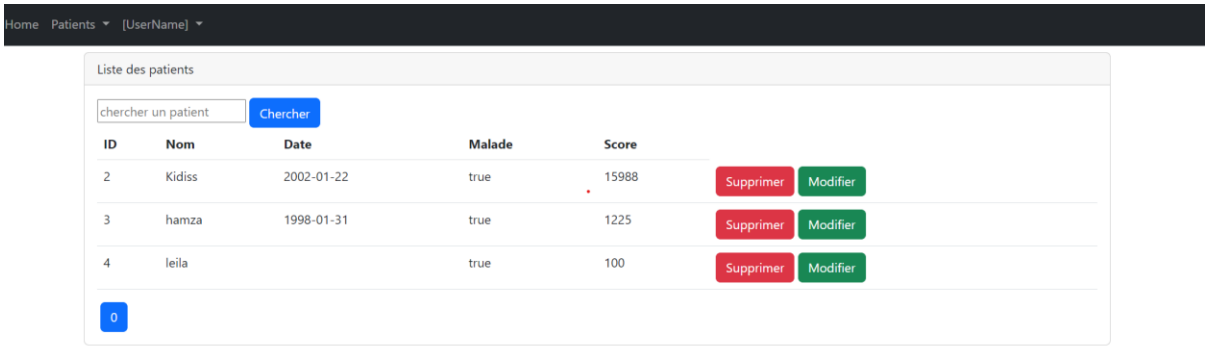


Couche DAO avec Spring data

1. On a créer un package Repository
2. On a declaré pour chaque entité une interface EntityRepository qui hérite del’interface générique JpaRepository

3. On a declarer les signatures des méthodes en respectant les règles de l'écriturepour récupérer les données sans avoir besoin de les implémenter grâce à Spring Boot

Couche Web



Maison Les patients [Nom d'utilisateur]

IDENTIFIANT 2

Nom

Kidiss

Date Naissance

22/01/2002

Malade

Score

15988

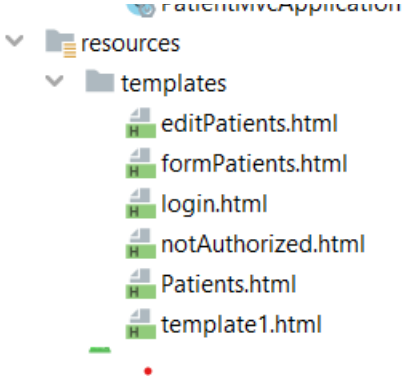
Sauvegarder

Controleurs

Une classe qui gère les requêtes http
La route ‘/user/index est liée à la méthode patients() elle va être appelée lorsqu’une requête de type GET est envoyée

Vues

On a travaillée avec le moteur de template thymeleaf



On ajout le dialect thyemeleaf

```
<html lang="en"
      xmlns:th="http://www.thymelef.org"
      xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"
>
```

Exemple d’une vue

```
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <link rel="stylesheet" href="/webjars/bootstrap/5.2.3/css/bootstrap.min.css">
    <script src="/webjars/bootstrap/5.2.3/js/bootstrap.bundle.js"></script>
</head>
<body>
<nav class="navbar navbar-expand-sm navbar-dark bg-dark">
    <div class="container-fluid">

        <div class="collapse navbar-collapse" id="mynavbar">
            <ul class="navbar-nav ">
                <li class="nav-item">
                    <a class="nav-link" th:href="@{index}">Home</a>
                </li>
                <li class="nav-item dropdown">
                    <a class="nav-link dropdown-toggle" href="#" role="button" data-bs-toggle="dropdown">
                        Patients
                    </a>
                    <ul class="dropdown-menu">
                        <li><a class="dropdown-item" th:href="@{formPatients}">Nouveau</a></li>
                        <li><a class="dropdown-item" th:href="@{index}">Cherche</a></li>
                    </ul>
                </li>
            </ul>
```

La methode save() reçoit les paramètres suivant: un objet de type Patient, quel page, sa taille et le keyword de la recherche

Authentification SPRING SECURITY

La méthode suivante prend comme paramètre HttpSecurity et va servir a spécifier les droits d’accès

```
no usages
@Bean
public InMemoryUserDetailsManager inMemoryUserDetailsManager(){
    return new InMemoryUserDetailsManager(
            User.withUsername("user1").password(passwordEncoder.encode( rawPassword: "1234")).roles("USER").build(),
            User.withUsername("user2").password(passwordEncoder.encode( rawPassword: "1234")).roles("USER").build(),
            User.withUsername("admin").password(passwordEncoder.encode( rawPassword: "1234")).roles("USER","ADMIN").build()
    );
}
```

```
no usages
@Bean
public SecurityFilterChain securityFilterChain(HttpSecurity httpSecurity)
        throws Exception {
    httpSecurity.formLogin();
    //.loginPage("/login").permitAll();
      httpSecurity.rememberMe();
    httpSecurity.authorizeHttpRequests().requestMatchers( ...patterns: "/webjars/** "," /h2-console/**").permitAll();
    httpSecurity.authorizeHttpRequests().requestMatchers( ...patterns: "/user/**").hasRole("USER");
    httpSecurity.authorizeHttpRequests().requestMatchers( ...patterns: "/admin/**").hasRole("ADMIN");
    httpSecurity.authorizeHttpRequests().anyRequest().authenticated();
    httpSecurity.exceptionHandling().accessDeniedPage( accessDeniedUrl: "/notAuthorized");
    return httpSecurity.build();
}
```