Activité Píatique Spíing MVC l'thymeleaf ,Spring Sécurité Gestion des Etudiants

Réaliser par :

Mohamed amine KHAMMOUR

Professeur :

Mr. Mohamed YOUSSFI

Enonce d’activité

Activité Pratique Spring MVC, Spring Data JPA, Spring Security

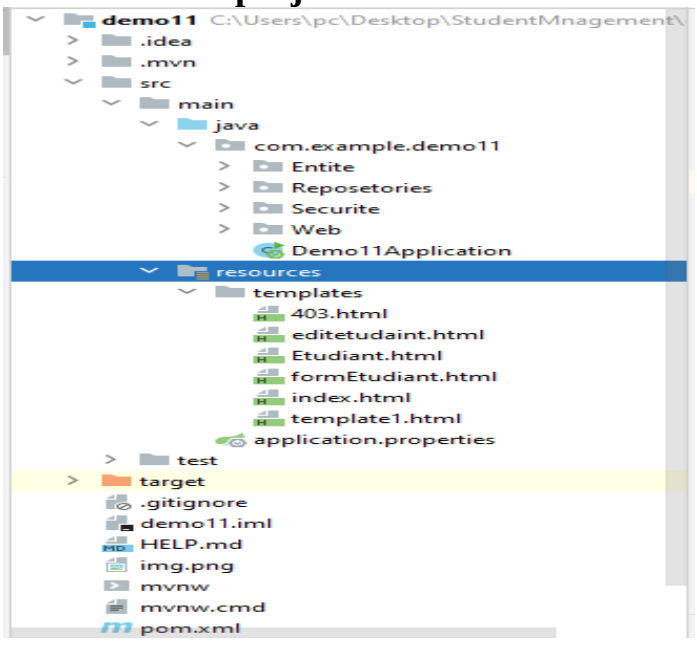
Mohamed YOUSSFI • 11 avr.

100 points

Créer une application Web basée sur Spring MVC, Spring Data JPA et Spring Security qui permet de gérer des étudiants.
Chaque étudiant est défini par:
- Son id
- Son nom
- Son prénom
- Son email
- Sa date naissance
- Son genre : MASCULIN ou FEMININ
- Un attribut qui indique si il est en règle ou non

L'application doit offrir les fonctionnalités suivantes :
- Chercher des étudiants par nom
- Faire la pagination
- Supprimer des étudiants en utilisant la méthode (DELETE au lieu de GET)
- Saisir et Ajouter des étudiants avec validation des formulaires
- Editer et mettre à jour des étudiants
- Créer une page template
- Sécuriser l'accès à l'application avec un système d'authentification basé sur Spring security en utilisant la stratégie UseDetails Service
- Ajouter d'autres fonctionnalités supplémentaires

Structure du projet :



La configuration du projet :

```
#spring.datasource.url=jdbc:h2:mem:PATIENT_DB
#spring.h2.console.enabled=true
server.port=8080
spring.datasource.url=jdbc:mysql://localhost:3306/TESO?createDatabaseIfNotExist=true
spring.datasource.username=root
spring.datasource.password=
spring.jpa.hibernate.ddl-auto = update
spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MariaDBDialect
spring.jpa.show-sql=false
spring.thymeleaf.cache=false
spring.main.allow-circular-references=true
```

La class Etudiant

```java
package com.example.demo11.Entite;

import ...
@Data
@AllArgsConstructor
@NoArgsConstructor
@Entity
public class Etudiant {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id ;
    @Size(min = 4,max = 50)
    private String nom ;
    private String Prenom ;
    @Temporal(TemporalType.DATE)
    @DateTimeFormat(pattern = "yyyy-MM-dd")
    private Date date;
    private Boolean EnRegle;
    private  StatueGenre Genre;
}
```

Les dépendances important du projet :

```xml
  <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>
  <dependency>
      <groupId>mysql</groupId>
      <artifactId>mysql-connector-java</artifactId>
  </dependency>
```

L'interface EtudiantReposetory : contient le prototype de la fonction qui permet de retourner unpatient par son nom :

```
import ...

public interface EtudiantReposetory extends JpaRepository<Etudiant,Long> {
    Page<Etudiant> findByNomContains(String nom, Pageable pageable);
}
```

le Reposetory Web qui contient la class controllerEtudiant :

```
public class ContrlerEtudiant {
    private EtudiantReposetory etudiantReposetory;

    @GET(path="/user/index")
    public String patiates(Model model ,
                           @Request Param(name ="page" ,defaultValue = "0") int page,
                           @Request Param(name = "size",defaultValue = "5") int size,
                           @Request Param(name = "keyword",defaultValue ="") String keyword)

    {
        Page<Etudiant> pagePatient= etudiantReposetory.findByNomContains(keyword,PageRequest.of(page,size));
        model.addAttribute( attributeName: "PageEudiant",pagePatient.getContent());
        model.addAttribute( attributeName: "nombrePages",new int[pagePatient.getTotalPages()]);
        model.addAttribute( attributeName: "currentPage",page);
        model.addAttribute( attributeName: "keyword",keyword);
        return "Etudiant.html";
    }
    @GET("/Admin/delete")
    public String delete(long id,String keyword,int page){
        etudiantReposetory.deleteById(id);
```

- **Dans le chemin `ressources/templates`, j'ai créé le template `Etudiant.html`,et d'autre templates est traité par le moteur de rendu thymeleaf, reçoit les données nécessaires à travers le controlleur.**
- **Dans ce template j'ai inclu les framework bootstrap et jquery qui sont stocké dans les ressources statiques.**

```
<table class="table">
<thead>
<tr>
    <th>Id</th><th>Nom</th><th>prenom</th><th>Date</th><th>EnRegle</th><th>Genre</th>
</tr>
</thead>
<tbody>
<tr th:each="p:${PageEudiant}">
    <td th:text="${p.id}"></td>
    <td th:text="${p.nom}"></td>
    <td th:text="${p.prenom}"></td>

    <td th:text="${p.date}"></td>
    <td th:text="${p.getEnRegle}"></td>
    <td th:text="${p.genre}"></td>
    <td sec:authorize="hasAuthority('ADMIN')">
    <a onclick="return confirm('Etes vous sure?')" class="btn btn-danger" th:href="@{/Admin/delete(i
        Delete
    </a>
    </td>
    <td sec:authorize="hasAuthority('ADMIN')">
        <a  class="btn btn-success" th:href="@{/Admin/editEtudaint(id=${p.id},keyword=${keyword},page
            Edite
        </a>
    </td>
```

La partie de sécurité en utilisant la stratégie UseDetails Service :

L'interface securite service :

```
import ...

public interface SecuriteService {
    AppUser saveNewUser(String username ,String password ,String verifyPasswor);
    AppRole savaNewRole(String roleName,String description);
    void addRoleToUser(String  username,String roleName);
    void removeRoleToUser(String  username,String roleName);
    AppUser loadUserBYUsername(String  usrname);
```

Extrais de l' implémentation de l' interface :

```
@Transactional
public class SecuriteServiceImpl implements SecuriteService {
    @Autowired
    private AppRoleRepositery appRoleRepositery;
    @Autowired
    private AppUserRepositery appUserRepositery;
private PasswordEncoder passwordEncoder;
    @Override
    public AppUser saveNewUser(String username, String password, String verifyPasswor) {
        if(!password.equals(verifyPasswor)) throw new RuntimeException("ERORE PASSWORD");
        String HachPasword=passwordEncoder.encode(password);
        AppUser appUser=new AppUser();
        appUser.setUserId(UUID.randomUUID().toString());
        appUser.setUsername(username);
        appUser.setPassword(HachPasword);
        appUser.setActive(true);
        AppUser savedAppUser=appUserRepositery.save(appUser);
        return savedAppUser;
    }
```

La class UserDetaileServiceImpl

```
@Service
@AllArgsConstructor
public class UserDetaileServiceImpl implements UserDetailsService {
    private  SecuriteService securiteService;
    @Override
    public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
        AppUser appUser=securiteService.loadUserBYUsername(username);
       /* Collection<GrantedAuthority>  authorities=new ArrayList();
        appUser.getAppRoles().forEach(role -> {
            SimpleGrantedAuthority authority=new SimpleGrantedAuthority((role.getRole()));
            authorities.add(authority);
        });*/
        Collection<GrantedAuthority> authorities1=
                appUser.getAppRoles() List<AppRole>
                        .stream() Stream<AppRole>
                        .map(role->new SimpleGrantedAuthority(role.getRole())) Stream<SimpleGrantedAuthority>
                        .collect(Collectors.toList());
        User user= new User(appUser.getUsername(),appUser.getPassword(),authorities1);
        return user;
    }
}
```

La class `securiteConfig`

```
@Configuration
@EnableWebSecurity
public class securiteConfig extends WebSecurityConfigurerAdapter {
↗ DataSource        @Autowired
    private DataSource dataSource;
↗ UserDetaileServiceImpl        @Autowired
    private UserDetaileServiceImpl userDetaileService2;
    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws Exception {
        auth.userDetailsService(userDetaileService2) ;
        /*
```

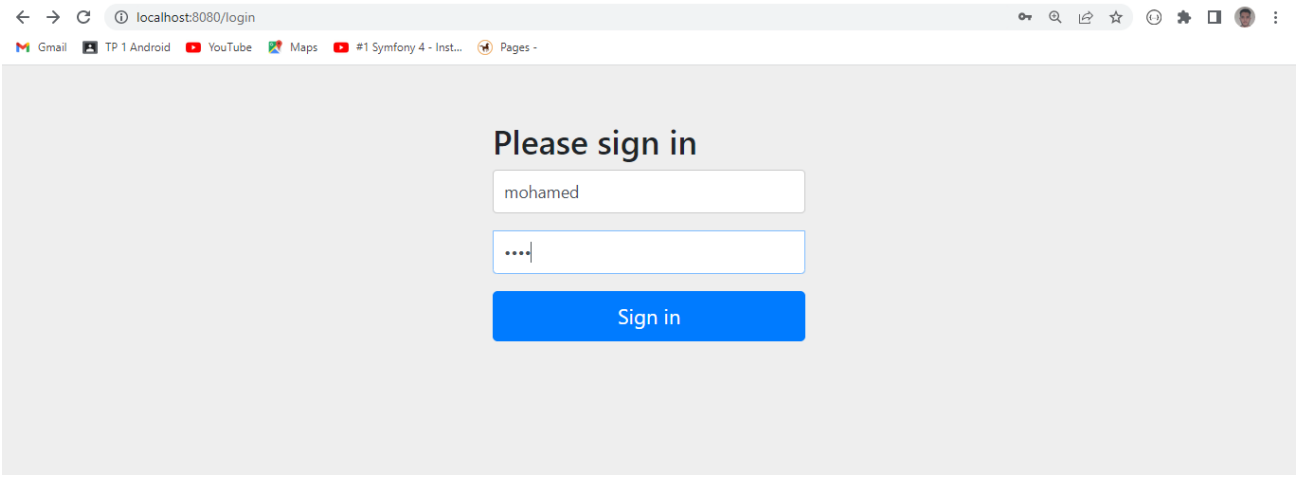
```
    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.authorizeRequests().antMatchers( ...antPatterns: "/Admin/**").hasAuthority("ADMIN");
        http.authorizeRequests().antMatchers( ...antPatterns: "/user/**").hasAuthority("USER");
        http.authorizeRequests().antMatchers( ...antPatterns: "/webjars/**", "/js/**", "/css/**", "/images/**").permitAll(
    http.authorizeRequests().antMatchers( ...antPatterns: "/").permitAll();

        http.formLogin();
        http.authorizeRequests().anyRequest().authenticated();
        http.exceptionHandling().accessDeniedPage("/403");



    }
    @Bean
    PasswordEncoder passwordEncoder() { return new BCryptPasswordEncoder(); }

}
```

La class application :

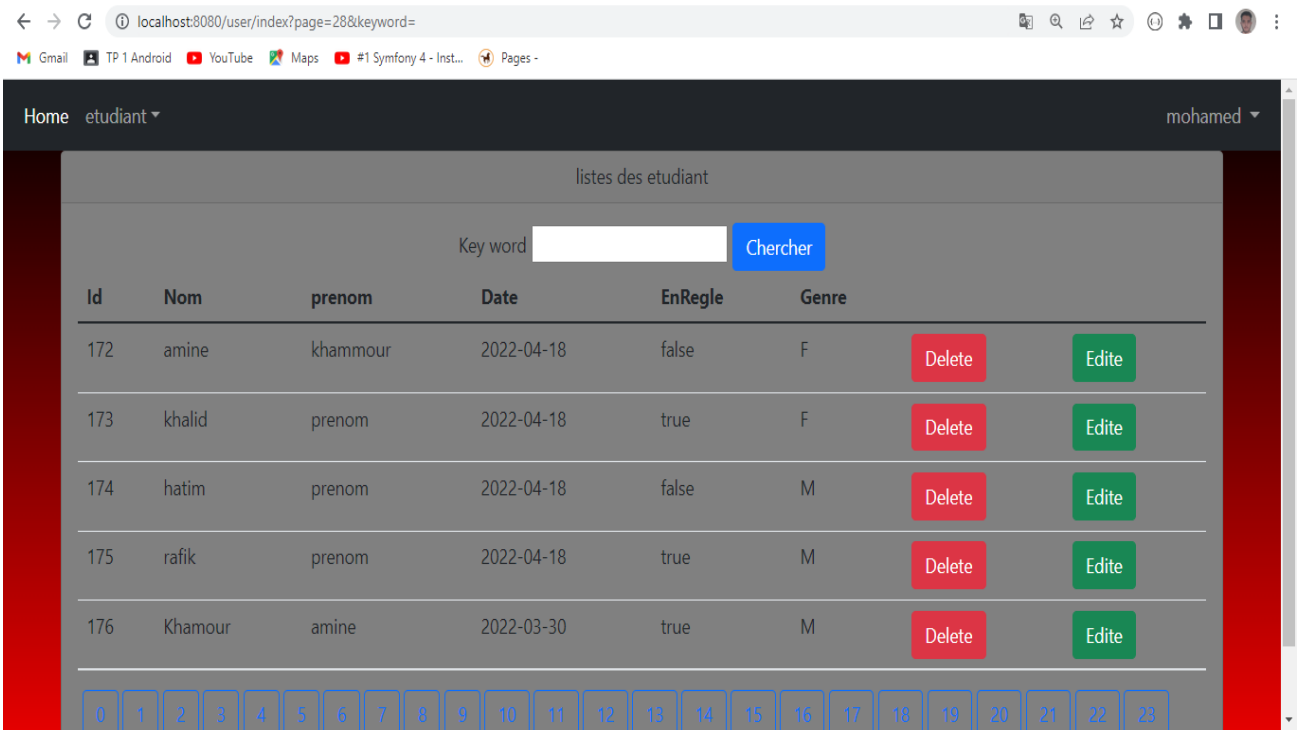
```
@Spring Boot Application
public class Demo11Application {

    public static void main(String[] args) { SpringApplication.run(Demo11Application.class, args); }

//@Bean
    CommandLineRunner commandLineRunner(EtudiantRepositery etudiantRepositery){
        return args -> {
            etudiantRepositery.save(new Etudiant( id: null, nom: "amine", Prenom: "khammour",new Date(), EnRegle: false, StatueGenre.F)) ;
            etudiantRepositery.save(new Etudiant( id: null, nom: "khalid", Prenom: "prenom",new Date(), EnRegle: true,StatueGenre.F)) ;
            etudiantRepositery.save(new Etudiant( id: null, nom: "hatim", Prenom: "prenom",new Date(), EnRegle: false,StatueGenre.M)) ;
            etudiantRepositery.save(new Etudiant( id: null, nom: "rafik", Prenom: "prenom",new Date(), EnRegle: true,StatueGenre.M)) ;

            etudiantRepositery.findAll().forEach(etudiant -> {
                System.out.println(etudiant.getNom());
            });

        };
}
```

```java
//@Bean
CommandLineRunner SaveUser(SecuriteService securiteService){
        return args -> {
          securiteService.saveNewUser( username: "mohamed", password: "1234", verifyPassword: "1234");
          securiteService.saveNewUser( username: "yassmin", password: "1234", verifyPassword: "1234");
          securiteService.saveNewUser( username: "hassan", password: "1234", verifyPassword: "1234");
          securiteService.savaNewRole( roleName: "USER", description: "");
          securiteService.savaNewRole( roleName: "ADMIN", description: "");
          securiteService.addRoleToUser( username: "mohamed", roleName: "USER");
            securiteService.addRoleToUser( username: "mohamed", roleName: "Admin");
        };

}

}
```

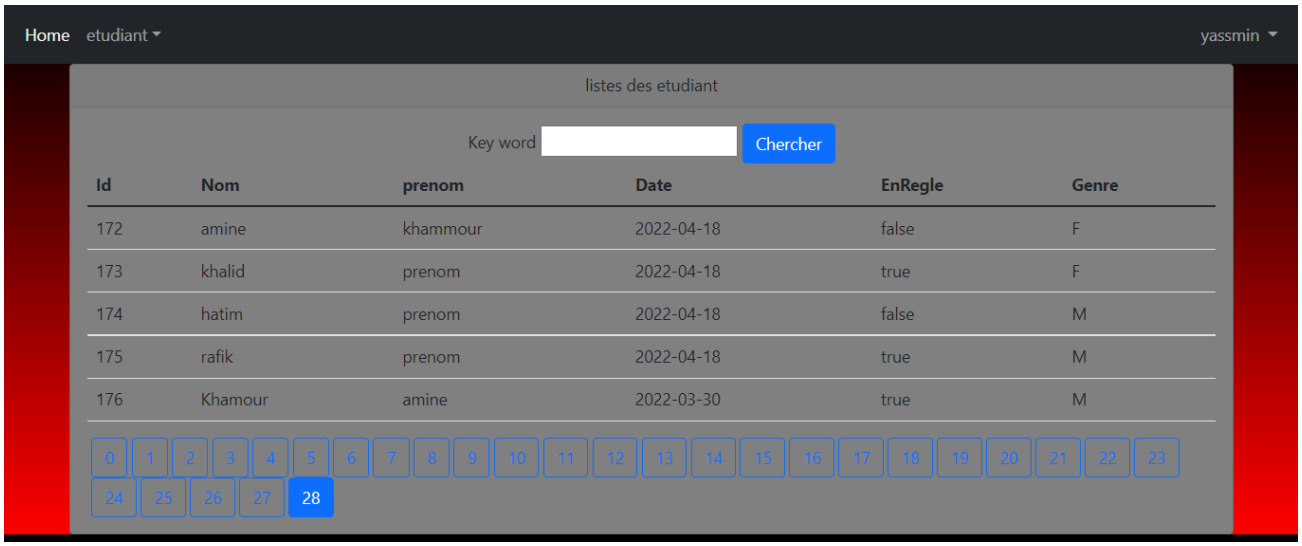
Résultat final :



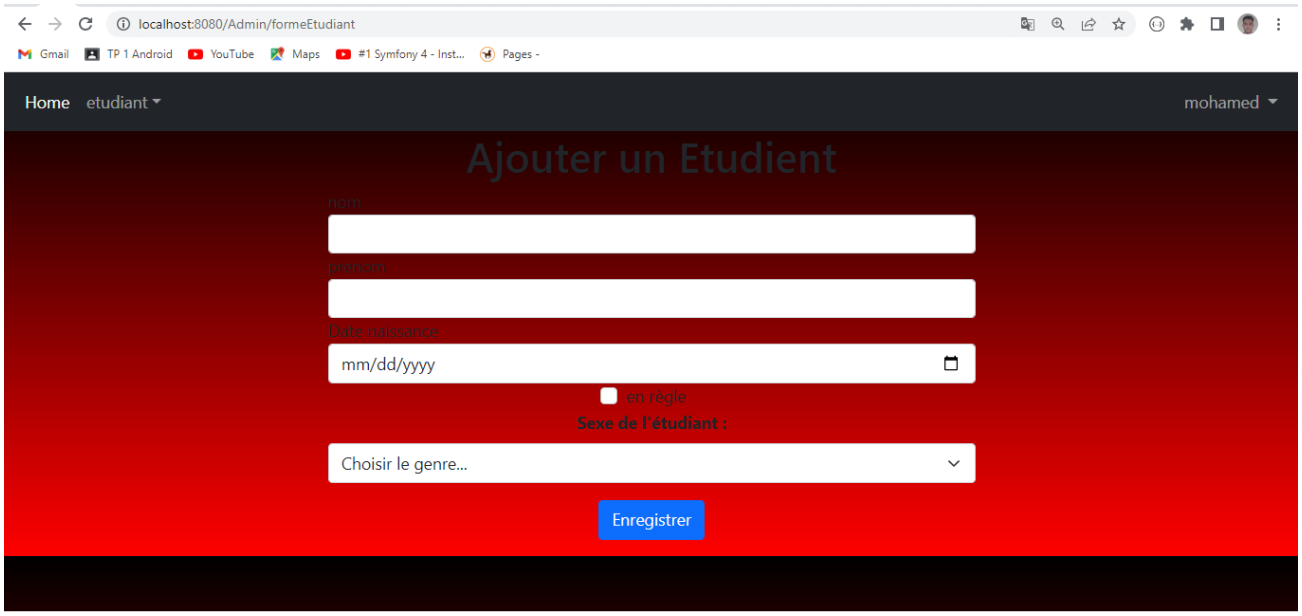
User Admin :



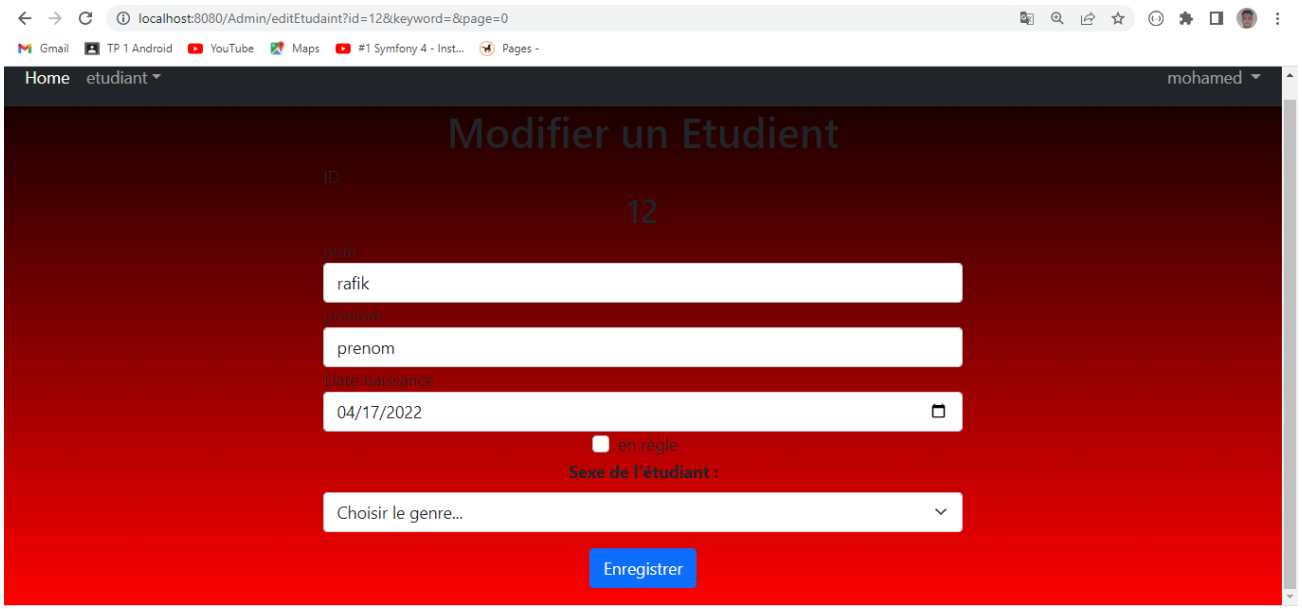
User Normal :



La partie d' ajout :



La partie modification :



La page d' erreurs :