Université Hassan II de Casablanca

DÉPARTEMENT MATHÉMATIQUES ET INFORMATIQUE

Activité Pratique Spring MVC, Spring Data JPA, Spring Security

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

JAVA JEE

Réalisé par :

RAMLI Abdelmajid

Année Universitaire: 2021-2022



Ecole Normale Supérieure de l'Enseignement **Technique Mohammedia**

Université Hassan II de Casablanca

Création des entités :

Entité Etudiant:

```
package com.example.demo.entities;
⊕ import java.util.Date;
 @Entity
 @Data @NoArgsConstructor @AllArgsConstructor
 public class Etudiant {
     @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
     private Long id;
     private String nom;
     private String prenom;
     private String email;
     @Temporal(TemporalType.DATE)
     @DateTimeFormat(pattern = "yyyy-MM-dd")
     private Date dateNaissance;
     private Genre genre;
     boolean enRegle;
 }
```

Repository Etudiant:

Nous allons utiliser cette méthode pour faire la pagination.

```
EtudiantMvcApplication.java
                              J Etudiant.java

    ■ EtudiantRepository.java ×
    package com.example.demo.repository;
 3⊕ import org.springframework.data.domain.Page;
10
11 @Repository
12 public interface EtudiantRepository extends JpaRepository<Etudiant, Long>{
13
         Page<Etudiant> findByNomContains(String keyword, Pageable pageable);
14
15 }
16
```

Pour la partie web nous allons crée les routes de l'application dans cette classe :

```
package com.example.demo.web;
3⊕ import org.springframework.data.domain.Page;
18
19 @Controller
20 @AllArgsConstructor
21 public class EtudiantController {
        private EtudiantRepository etudiantRepository;
22
        @GetMapping( "/")
24⊖
        public String home(){
25
26
            return "home";
27
28
        @GetMapping( "/403")
29⊖
30
       public String error(){
            return "403";
31
32
33
349
        @GetMapping(path="/user/etudiants")
35
        public String patients(Model model,
                                @RequestParam(name = "page", defaultValue = "0") int page,
36
                                @RequestParam(name = "size", defaultValue = "6") int size,
37
                                @RequestParam(name = "keyword", defaultValue = "") String keyword) {
38
            Page<Etudiant> etudiantsPage=etudiantRepository.findByNomContains(keyword,PageRequest.of(page, size));
39
40
            model.addAttribute("listEtudiants",etudiantsPage.getContent());
            model.addAttribute("pages",new int[etudiantsPage.getTotalPages()]);
model.addAttribute("currentPage",page);
41
42
            model.addAttribute("keyword", keyword);
43
            return "etudiants";
44
        }
45
```

```
@GetMapping(path="/admin/editEtudiant")
public String editPatient(Model model,Long id,int page,String keyword) {
    Etudiant etudiant=etudiantRepository.findById(id).orElse(null);
    if(etudiant==null) {
    throw new RuntimeException("etudiant n'existe pas");
    model.addAttribute("etudiant",etudiant);
    model.addAttribute("page",page);
   model.addAttribute("keyword", keyword);
    return "editEtudiant";
}
@GetMapping(path="/user/showEtudiant")
public String showPatient(Model model,Long id,int page,String keyword) {
    Etudiant etudiant=etudiantRepository.findById(id).orElse(null);
    if(etudiant==null) {
    throw new RuntimeException("etudiant n'existe pas");
    model.addAttribute("etudiant",etudiant);
    model.addAttribute("page",page);
   model.addAttribute("keyword", keyword);
    return "showEtudiant";
}
```

Université Hassan II de Casablanca

Nous allons utiliser Thymeleaf pour générer les pages html :

Nous avons utilisé une page template :

```
<!DOCTYPE html>
 2⊖ <html xmlns:th="http://www.thymeleaf.org"
          xmlns:layout="http://www.utraq.net.nz/thymeleaf/layout"
          xmlns:sec="http://www.thymeleaf.org/extras/spring-security">
59 <head>
6 <meta charset="UTF-8">
7 <title>Insert title here</title>
8 k rel="stylesheet" type="text/css" href="/webjars/bootstrap/5.1.3/css/bootstrap.min.css">
9 <script src="/webjars/bootstrap/5.1.3/js/bootstrap.bundle.js"></script>
10 </head>
11<sup>⊕</sup> <body>
12⊖ <nav class="navbar navbar-expand-sm" style="background-color: ■ tomato;">
13⊖ <div class="container-fluid">
       <a class="navbar-brand" href="javascript:void(0)"style="color: white;">Ramli</a>
<button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#mynavbar">
14
15⊖
         <span class="navbar-toggler-icon"></span>
16
17
        </button>
18⊕
       <div class="collapse navbar-collapse" id="mynavbar">...
48
       </div>
     </div>
49
50 </nav>
51
52 <section layout:fragment="content1"></section>
53
54 </body>
55 </html>
```

Pour l'affichage des étudiants, nous avons récupérer la liste des étudiants, et a l'aide d'une boucle for each, thymeleaf va parcourir la liste et afficher les informations de chaque élément de la liste :

Ecole Normale Supérieure de l'Enseignement Technique Mohammedia

Université Hassan II de Casablanca

Modifier un étudiant :

Dans ce fichier, on récupère et on affiche les informations d'un étudiant, pour le modifier par la suite :

```
<!DOCTYPE html>
           xmlns:th="http://www.thymeleaf.org"
 2⊖ <html
           xmlns:layout="http://www.utraq.net.nz/thymeleaf/layout"
           layout:decorate="template">
4
5⊖ <head>
6 <meta charset="UTF-8">
7 <title>Ajouter</title>
8 k rel="stylesheet" href="/webjars/bootstrap/5.1.3/css/bootstrap.min.css">
10⊖ <body>
11⊖ <div layout:fragment="content1">
12⊖ <div class="container">
13@<form th:action="@{/admin/save(page=${page},keyword=${keyword})}" method="post">
149
       <div>
15
           <label>ID</label>
           <label th:text="${etudiant.id}"></label>
16
           <input class="form-control" type="hidden" name="id" th:value="${etudiant.id}">
17
       </div>
18
19@
       <div>
           <label>Nom</label>
20
           <input class="form-control" type="text" name="nom" th:value="${etudiant.nom}">
21
22
           <span th:errors="${etudiant.nom}"></span>
       </div>
23
24⊖
           <div>
25
           <label>Prenom</label>
           <input class="form-control" type="text" name="prenom" th:value="${etudiant.prenom}">
26
           <span th:errors="${etudiant.prenom}"></span>
27
28
       </div>
29⊖
           <div>
30
           <label>Email</label>
           <input class="form-control" type="text" name="email" th:value="${etudiant.email}">
31
           <span th:errors="${etudiant.email}"></span>
32
       </div>
33
```

```
<label>Genre</label>
45
46
            <input class="form-control" type="text" name="genre" th:value="${etudiant.genre}">
            <span th:errors="${etudiant.genre}"></span>
47
48
       </div>
       <button type="submit" class="btn btn-primary">Save</button>
49
50 </form>
51
52
53 </div>
54
55
   </div>
56
57
  </body>
   </html>
```



Université Hassan II de Casablanca

form etudiant:

Dans ce fichier, on effectuer l'ajout d'un nouveau étudiant :

```
<!DOCTYPE html>
           xmlns:th="http://www.thymeleaf.org"
2⊖ <html
           xmlns:layout="http://www.utraq.net.nz/thymeleaf/layout"
           layout:decorate="template">
5⊖ <head>
6 <meta charset="UTF-8">
 <title>Ajouter</title>
8 <link rel="stylesheet" href="/webjars/bootstrap/5.1.3/css/bootstrap.min.css">
9 </head>
0⊖ <body>
1⊖ <div layout:fragment="content1">
2<sup>©</sup> <div class="container">
3<sup>©</sup> <form th:action="@{/admin/save}" method="post">
5⊝
           <label>Nom</label>
6
           <input class="form-control" type="text" name="nom" th:value="${etudiant.nom}">
           <span th:errors="${etudiant.nom}"></span>
8
9
       </div>
00
           <div>
           <label>Prenom</label>
2
           <input class="form-control" type="text" name="prenom" th:value="${etudiant.prenom}">
3
           <span th:errors="${etudiant.prenom}"></span>
4
       </div>
5⊝
           <div>
           <label>Email</label>
6
           <input class="form-control" type="text" name="email" th:value="${etudiant.email}">
8
           <span th:errors="${etudiant.email}"></span>
9
       </div>
:00
       <div>
           <label>Date Naissance</label>
           <input class="form-control" type="date" name="dateNaissance" th:value="${etudiant.dateNaissance}">
           <span th:errors="${etudiant.dateNaissance}"></span>
```

```
<div>
30€
31
           <label>Date Naissance</label>
            <input class="form-control" type="date" name="dateNaissance" th:value="${etudiant.dateNaissance}">
32
            <span th:errors="${etudiant.dateNaissance}"></span>
33
       </div>
34
35⊖
           <label>EnRegle</label>
36
           <input type="checkbox" name="enRegle" th:checked="${etudiant.enRegle}">
37
            <span th:errors="${etudiant.enRegle}"></span>
38
       </div>
39
400
       <div>
           <label>Genre</label>
41
           <input class="form-control" type="text" name="genre" th:value="${etudiant.genre}">
42
           <span th:errors="${etudiant.genre}"></span>
43
       </div>
       <button type="submit" class="btn btn-primary">Save</button>
45
46 </form>
47
48
49 </div>
50
   </div>
51
52
53 </body>
   </html>
```



Pour sécuriser l'accès a l'application, nous avons utiliser spring security avec la stratégie user details :

```
package com.example.demo.security;
₃ 3⊕ import org.springframework.beans.factory.annotation.Autowired;
13
14 @Configuration
15 @EnableWebSecurity
16 public class SecurityConfiguration extends WebSecurityConfigurerAdapter {
17
18⊖
        UserDetailsService userDetailsService;
19
20
21⊖
        @Override
22
        protected void configure(AuthenticationManagerBuilder auth) throws Exception {
            auth.userDetailsService(userDetailsService);
24
        }
25
26
27⊖
        @Bean
28
        public PasswordEncoder getPasswordEncoder() {
            return NoOpPasswordEncoder.getInstance();
129
30
31
32⊖
        @Override
33
        protected void configure(HttpSecurity http) throws Exception {
            http.authorizeRequests().antMatchers("/admin/**").hasRole("ADMIN")
34
            .antMatchers("/user/**").hasAnyRole("USER","ADMIN")
35
            .antMatchers("/webjars/**").permitAll()
36
            .antMatchers("/").permitAll()
37
            .and().formLogin();
38
            http.exceptionHandling().accessDeniedPage("/403");
39
        }
40
41
```

App user details service:

```
package com.example.demo.security;
3⊕ import java.util.ArrayList;
8 @Service
  public class AppUserDetailsService implements UserDetailsService{
19
00
       @Autowired
       AppUserRepository appUserRepository;
7
23⊝
       @Override
4
       public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
           AppUser appUser= appUserRepository.findByUsername(username);
25
26
           AppUserDetails appUserDetails=new AppUserDetails(appUser);
           return appUserDetails;
89
       }
29
  }
80
```



Dans ce fichier, on renvoie un objet de type user details avec le username, le password, et une liste des rôles :

```
package com.example.demo.security;
3⊕ import java.util.ArrayList;
  public class AppUserDetails implements UserDetails {
5
       private String userName;
6
       private String userPassword;
7
       private List<GrantedAuthority> authorities;
8
90
       public AppUserDetails(AppUser user) {
20
           System.out.println(user.getUsername()+" "+user.getPassword());
1
           userName=user.getUsername();
22
           userPassword=user.getPassword();
23
           authorities = new ArrayList<>();
4
           for(AppRole role:user.getAppRoles()) {
25
               authorities.add(new SimpleGrantedAuthority(role.getRoleName()));
26
           //authorities.add(new SimpleGrantedAuthority("ROLE ADMIN"));
28
       }
99
       @Override
       public Collection<? extends GrantedAuthority> getAuthorities() {
80
31
32
           return authorities;
33
       }
34
35€
       @Override
       public String getPassword() {
36
           return userPassword;
88
       }
39
```

```
419
        public String getUsername() {
42
43
44
             return userName;
45
46
47⊖
        @Override
        public boolean isAccountNonExpired() {
48
49
50
             return true;
51
52
53⊖
54
        public boolean isAccountNonLocked() {
55
56
57
58
59<sup>⊕</sup>
             return true;
        @Override
        public boolean isCredentialsNonExpired() {
60
61
62
            return true;
63
64
65⊖
        @Override
        public boolean isEnabled() {
66
67
             return true;
68
69
70
```

علضيبلا رادلا ياثلا نسحلا قعماج



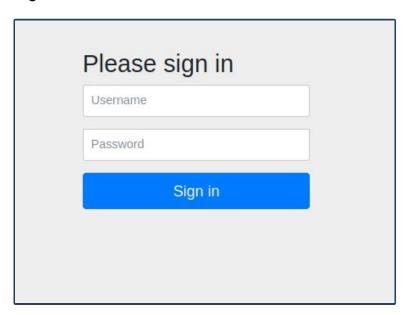
Ecole Normale Supérieure de l'Enseignement Technique Mohammedia

Université Hassan II de Casablanca

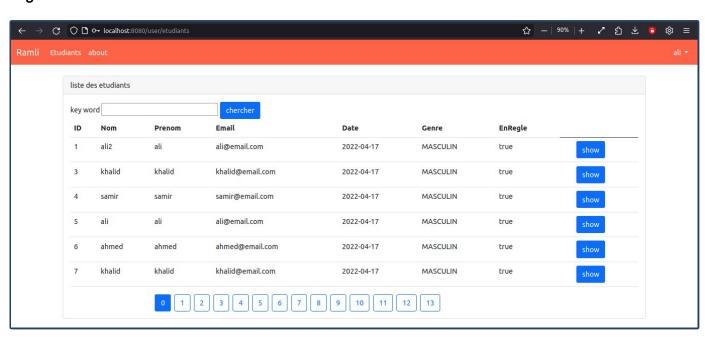
Démo de l'application :

Authentification avec utilisateur ayant le rôle « USER » :

Login:



Page étudiants :



علضيبلا رادلا يناثلا نسحلا قعماج





Ecole Normale Supérieure de l'Enseignement Technique Mohammedia

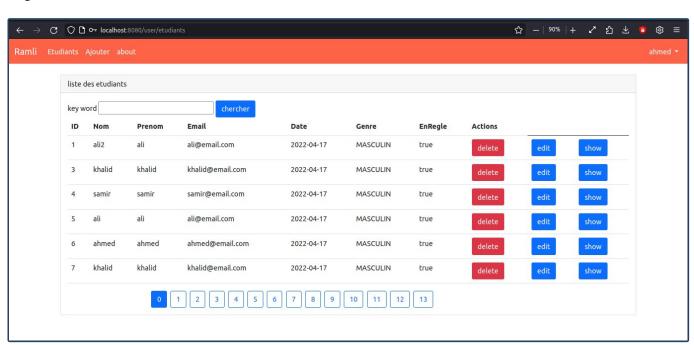
Université Hassan II de Casablanca

Avec utilisateur ayant le rôle « ADMIN » :

Login:



Page étudiants :



On peut voir que l'utilisateur avec le rôle ADMIN a le droit d'ajouter de modifier et de supprimer un étudiant .

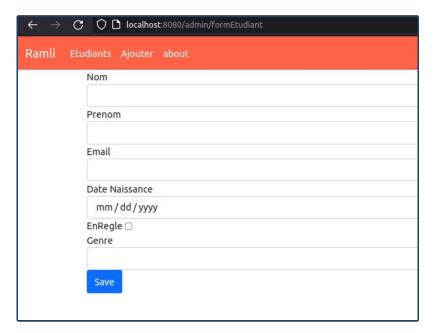
عاضيباا رادلا ياثالا نسحاا قعماج



Ecole Normale Supérieure de l'Enseignement Technique Mohammedia

Université Hassan II de Casablanca

Ajouter un étudiant :



Modifier un étudiant :



Supprimer un étudiant:

