

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

Création des entités :

Entité Etudiant :

```
1 package com.example.demo.entities;
2
3 + import java.util.Date;
4
5 @Entity
6 @Data @NoArgsConstructor @AllArgsConstructor
7 public class Etudiant {
8     @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
9     private Long id;
10
11     private String nom;
12     private String prenom;
13     private String email;
14
15     @Temporal(TemporalType.DATE)
16     @DateTimeFormat(pattern = "yyyy-MM-dd")
17     private Date dateNaissance;
18
19     private Genre genre;
20
21     boolean enRegle;
22 }
23 }
```

Repository Etudiant :

Nous allons utiliser cette méthode pour faire la pagination.

```
EtudiantMvcApplication.java Etudiant.java EtudiantRepository.java X
1 package com.example.demo.repository;
2
3 + import org.springframework.data.domain.Page;
4
5 @Repository
6 public interface EtudiantRepository extends JpaRepository<Etudiant, Long>{
7
8     Page<Etudiant> findByNomContains(String keyword,Pageable pageable);
9 }
10
11
12
13
14
15
16
```

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

```
1 package com.example.demo.web;
2
3 import org.springframework.data.domain.Page;
4
18 @Controller
19 @AllArgsConstructor
20 public class EtudiantController {
21     private EtudiantRepository etudiantRepository;
22
23
24 @GetMapping("/")
25 public String home(){
26     return "home";
27 }
28
29 @GetMapping("/403")
30 public String error(){
31     return "403";
32 }
33
34 @GetMapping(path="/user/etudiants")
35 public String patients(Model model,
36     @RequestParam(name = "page",defaultValue = "0") int page,
37     @RequestParam(name = "size",defaultValue = "6") int size,
38     @RequestParam(name = "keyword",defaultValue = "") String keyword) {
39     Page<Etudiant> etudiantsPage=etudiantRepository.findByNomContains(keyword,PageRequest.of(page, size));
40     model.addAttribute("listEtudiants",etudiantsPage.getContent());
41     model.addAttribute("pages",new int[etudiantsPage.getTotalPages()]);
42     model.addAttribute("currentPage",page);
43     model.addAttribute("keyword",keyword);
44     return "etudiants";
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";
}
```

Nous allons utiliser Thymeleaf pour générer les pages html :

Nous avons utilisé une page template :

```
1 <!DOCTYPE html>
2 <html xmlns:th="http://www.thymeleaf.org"
3     xmlns:layout="http://www.utraq.net.nz/thymeleaf/layout"
4     xmlns:sec="http://www.thymeleaf.org/extras/spring-security">
5 <head>
6 <meta charset="UTF-8">
7 <title>Insert title here</title>
8 <link 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 <body>
12 <nav class="navbar navbar-expand-sm" style="background-color: # tomato;">
13 <div class="container-fluid">
14 <a class="navbar-brand" href="javascript:void(0)" style="color: white;">Ramli</a>
15 <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#mynavbar">
16 <span class="navbar-toggler-icon"></span>
17 </button>
18 <div class="collapse navbar-collapse" id="mynavbar">
19 <div>
20 </div>
21 </div>
22 </nav>
23 <section layout:fragment="content1"></section>
24 </body>
25 </html>
```

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

```
<tr th:each="p:${listEtudiants}">
  <td th:text="${p.id}"></td>
  <td th:text="${p.nom}"></td>
  <td th:text="${p.prenom}"></td>
  <td th:text="${p.email}"></td>
  <td th:text="${p.dateNaissance}"></td>
  <td th:text="${p.genre}"></td>
  <td th:text="${p.enRegle}"></td>
```


Modifier un étudiant :

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

```
1 <!DOCTYPE html>
2 <html xmlns:th="http://www.thymeleaf.org"
3       xmlns:layout="http://www.utraq.net.nz/thymeleaf/layout"
4       layout:decorate="template">
5 <head>
6 <meta charset="UTF-8">
7 <title>Ajouter</title>
8 <link rel="stylesheet" href="/webjars/bootstrap/5.1.3/css/bootstrap.min.css">
9 </head>
10 <body>
11 <div layout:fragment="content1">
12 <div class="container">
13 <form th:action="@{/admin/save(page=${page}, keyword=${keyword})}" method="post">
14 <div>
15 <label>ID</label>
16 <input th:text="${etudiant.id}" type="text"/>
17 <input class="form-control" type="hidden" name="id" th:value="${etudiant.id}">
18 </div>
19 <div>
20 <label>Nom</label>
21 <input class="form-control" type="text" name="nom" th:value="${etudiant.nom}">
22 <span th:errors="${etudiant.nom}"></span>
23 </div>
24 <div>
25 <label>Prenom</label>
26 <input class="form-control" type="text" name="prenom" th:value="${etudiant.prenom}">
27 <span th:errors="${etudiant.prenom}"></span>
28 </div>
29 <div>
30 <label>Email</label>
31 <input class="form-control" type="text" name="email" th:value="${etudiant.email}">
32 <span th:errors="${etudiant.email}"></span>
33 </div>
34 <div>
35 <label>Genre</label>
36 <input class="form-control" type="text" name="genre" th:value="${etudiant.genre}">
37 <span th:errors="${etudiant.genre}"></span>
38 </div>
39 <button type="submit" class="btn btn-primary">Save</button>
40 </form>
41 </div>
42 </div>
43 </div>
44 </body>
45 </html>
```

form etudiant :

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

```
1 <!DOCTYPE html>
2 <html xmlns:th="http://www.thymeleaf.org"
3     xmlns:layout="http://www.utraq.net.nz/thymeleaf/layout"
4     layout:decorate="template">
5 <head>
6 <meta charset="UTF-8">
7 <title>Ajouter</title>
8 <link rel="stylesheet" href="/webjars/bootstrap/5.1.3/css/bootstrap.min.css">
9 </head>
10 <body>
11 <div layout:fragment="content1">
12 <div class="container">
13 <form th:action="@{/admin/save}" method="post">
14
15     <div>
16         <label>Nom</label>
17         <input class="form-control" type="text" name="nom" th:value="${etudiant.nom}">
18         <span th:errors="${etudiant.nom}"></span>
19     </div>
20     <div>
21         <label>Prenom</label>
22         <input class="form-control" type="text" name="prenom" th:value="${etudiant.prenom}">
23         <span th:errors="${etudiant.prenom}"></span>
24     </div>
25     <div>
26         <label>Email</label>
27         <input class="form-control" type="text" name="email" th:value="${etudiant.email}">
28         <span th:errors="${etudiant.email}"></span>
29     </div>
30     <div>
31         <label>Date Naissance</label>
32         <input class="form-control" type="date" name="dateNaissance" th:value="${etudiant.dateNaissance}">
33         <span th:errors="${etudiant.dateNaissance}"></span>
34     </div>
35     <div>
36         <label>EnRegle</label>
37         <input type="checkbox" name="enRegle" th:checked="${etudiant.enRegle}">
38         <span th:errors="${etudiant.enRegle}"></span>
39     </div>
40     <div>
41         <label>Genre</label>
42         <input class="form-control" type="text" name="genre" th:value="${etudiant.genre}">
43         <span th:errors="${etudiant.genre}"></span>
44     </div>
45     <button type="submit" class="btn btn-primary">Save</button>
46 </form>
47
48 </div>
49 </div>
50
51 </div>
52
53 </body>
54 </html>
```

Pour sécuriser l'accès à l'application, nous avons utilisé Spring Security avec la stratégie user details :

```
1 package com.example.demo.security;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4
13 @Configuration
14 @EnableWebSecurity
15 public class SecurityConfiguration extends WebSecurityConfigurerAdapter {
16
17     @Autowired
18     UserDetailsService userDetailsService;
19
20
21     @Override
22     protected void configure(AuthenticationManagerBuilder auth) throws Exception {
23
24         auth.userDetailsService(userDetailsService);
25     }
26
27     @Bean
28     public PasswordEncoder getPasswordEncoder() {
29         return NoOpPasswordEncoder.getInstance();
30     }
31
32     @Override
33     protected void configure(HttpSecurity http) throws Exception {
34         http.authorizeRequests().antMatchers("/admin/**").hasRole("ADMIN")
35             .antMatchers("/user/**").hasAnyRole("USER", "ADMIN")
36             .antMatchers("/webjars/**").permitAll()
37             .antMatchers("/").permitAll()
38             .and().formLogin();
39         http.exceptionHandling().accessDeniedPage("/403");
40     }
41 }
```

App user details service :

```
1 package com.example.demo.security;
2
3 import java.util.ArrayList;
4
17 @Service
18 public class AppUserDetailsService implements UserDetailsService {
19
20     @Autowired
21     AppUserRepository appUserRepository;
22
23     @Override
24     public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
25         AppUser appUser = appUserRepository.findByUsername(username);
26         AppUserDetails appUserDetails = new AppUserDetails(appUser);
27         return appUserDetails;
28     }
29 }
30
31 }
```


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

```
1 package com.example.demo.security;
2
3 import java.util.ArrayList;
4
5 public class AppUserDetails implements UserDetails {
6     private String userName;
7     private String userPassword;
8     private List<GrantedAuthority> authorities;
9
10    public AppUserDetails(AppUser user) {
11        System.out.println(user.getUsername()+" "+user.getPassword());
12        userName=user.getUsername();
13        userPassword=user.getPassword();
14        authorities = new ArrayList<>();
15        for(AppRole role:user.getAppRoles()) {
16            authorities.add(new SimpleGrantedAuthority(role.getRoleName()));
17        }
18        //authorities.add(new SimpleGrantedAuthority("ROLE_ADMIN"));
19    }
20    @Override
21    public Collection<? extends GrantedAuthority> getAuthorities() {
22
23        return authorities;
24    }
25
26    @Override
27    public String getPassword() {
28
29        return userPassword;
30    }
31 }
```

```
41 @Override
42 public String getUsername() {
43
44     return userName;
45 }
46
47 @Override
48 public boolean isAccountNonExpired() {
49
50     return true;
51 }
52
53 @Override
54 public boolean isAccountNonLocked() {
55
56     return true;
57 }
58
59 @Override
60 public boolean isCredentialsNonExpired() {
61
62     return true;
63 }
64
65 @Override
66 public boolean isEnabled() {
67
68     return true;
69 }
70
71 }
```


Démo de l'application :

Authentification avec utilisateur ayant le rôle « USER » :

Login :

Please sign in

Sign in

Page étudiants :

localhost:8080/user/etudiants

Ramli Etudiants about ali

liste des etudiants

key word

chercher

ID	Nom	Prenom	Email	Date	Genre	EnRegle	
1	ali2	ali	ali@email.com	2022-04-17	MASCULIN	true	show
3	khalid	khalid	khalid@email.com	2022-04-17	MASCULIN	true	show
4	samir	samir	samir@email.com	2022-04-17	MASCULIN	true	show
5	ali	ali	ali@email.com	2022-04-17	MASCULIN	true	show
6	ahmed	ahmed	ahmed@email.com	2022-04-17	MASCULIN	true	show
7	khalid	khalid	khalid@email.com	2022-04-17	MASCULIN	true	show

0

1

2

3

4

5

6

7

8

9

10

11

12

13

Avec utilisateur ayant le rôle « ADMIN » :

Login :

Please sign in

You have been signed out

ahmed

.....|

Sign in

Page étudiants :

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

Ajouter un étudiant :

The screenshot shows a web browser window with the URL `localhost:8080/admin/FormEtudiant`. The page has a navigation bar with links: **Ramli**, **Etudiants**, **Ajouter**, and **about**. The form contains the following fields: **Nom** (text input), **Prenom** (text input), **Email** (text input), **Date Naissance** (date picker showing `mm / dd / yyyy`), **EnRegle** (checkbox), and **Genre** (text input). A blue **Save** button is at the bottom.

Modifier un étudiant :

The screenshot shows a web browser window with the URL `localhost:8080/admin/FormEtudiant`. The page has a navigation bar with links: **Ramli**, **Etudiants**, **Ajouter**, and **about**. The form contains the following fields: **ID 1** (text input), **Nom** (text input with value `ali2`), **Prenom** (text input with value `ali`), **Email** (text input with value `ali@email.com`), **Date Naissance** (date picker showing `04 / 17 / 2022`), **EnRegle** (checkbox with value `checked`), and **Genre** (text input with value `MASCULIN`). A blue **Save** button is at the bottom.

Supprimer un étudiant :

The screenshot shows a confirmation dialog box with the title `delete etudiant?`. The dialog has a dark background and a light border. It contains a globe icon and the text `localhost:8080`. At the bottom, there are two buttons: **Cancel** and **OK**.