Ecole Normale Supérieure de l'Enseignement Technique Mohammedia

Université Hassan II de Casablanca

ENSET

ا لمدرسة العليا لأساتذة التعليم التقني المحمدية

جامعة الحسن الثاني بالدار البيضاء

DEPARTEMENT MATHEMATIQUES ET INFORMATIQUE

Compte rendu

- Travaux Pratique : 4-

-

« Architecture JEE »

SPRING
MVC THYMELEAF
<<ETUDIANT>>

Réalisé par :

Ikram Berradi

Encadré par :

Mohamed YOUSSFI

GLSID II-2022/2021

❖ Exercice : application Web JEE basée sur Spring MVC, Thylemeaf et Spring Data JPA

Création d'une application Web JEE basée sur Spring MVC, Thylemeaf et Spring Data JPA qui permet de gérer les étudiants. L'application doit permettre les fonctionnalitéssuivantes :

- ✓ **Afficher les étudiants**
- ✓ **Faire la pagination**
- ✓ **Chercher les étudiants**
- ✓ **Supprimer un étudiant**
- ✓ **Faire des améliorations supplémentaires**

➢ Application.properties

application.properties

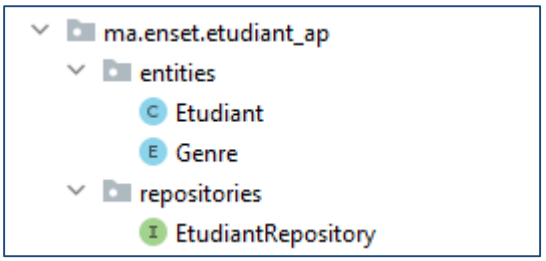
```
spring.datasource.url=jdbc:mysql://localhost:3306/etudiants_bd?createDatabaseIfNotExist=true
spring.datasource.username=root
spring.datasource.password=
server.port=8087
spring.jpa.hibernate.ddl-auto=update
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MariaDBDialect
spring.jpa.show-sql=true
```

➢ Application Spring Boot

EtudiantApApplication.java

```
@SpringBootApplication
public class EtudiantApApplication {

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

    @Bean
    PasswordEncoder passwordEncoder() { return new BCryptPasswordEncoder(); }
    // @Bean
    CommandLineRunner start(EtudiantRepository etudiantRepository){
        return args -> {
            etudiantRepository.save(
                    new Etudiant( id: null,  nom: "berradi", prenom: "ikram", email: "ikram@gmail.com", Date.valueOf("2000-01-27"), regle
            etudiantRepository.save(
                    new Etudiant( id: null,  nom: "badaoui", prenom: "sara", email: "sara@gmail.com", Date.valueOf("2001-02-07"), regle: f
            etudiantRepository.save(
                    new Etudiant( id: null,  nom: "alami", prenom: "salma", email: "salma@gmail.com", Date.valueOf("2000-04-20"),  regle:
            etudiantRepository.save(
                    new Etudiant( id: null,  nom: "alaoui", prenom: "Mohammed", email: "mohammed@gmail.com", Date.valueOf("2000-01-22"),

            etudiantRepository.findAll().forEach(e-> System.out.println("Etudiant ==> "+e.getNom()+" "+e.getPrenom()));
        };
    }
    // @Bean
    CommandLineRunner saveUsers(ISecurityService iSecurityService){
        return args -> {
```

➢ Couche DAO



➢ Entité JPA : Patient

Etudiant.java

```java
package ma.enset.etudiant_ap.entities;

import ...

@Entity
@Data @NoArgsConstructor @AllArgsConstructor
public class Etudiant {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @NotEmpty
    @Size(min = 4, max = 40)
    private String nom;
    @NotEmpty
    @Size(min = 4, max = 40)
    private String prenom;
    @Column(unique = true)
    @NotEmpty
    private String email;
    @DateTimeFormat(pattern = "yyyy-MM-dd")
    @Temporal(TemporalType.DATE)
    private Date dateNaissance;
    private boolean regle;
    @Enumerated(EnumType.STRING)
    private Genre genre;

```

➢ Interfaces DAO basées sur Spring Data

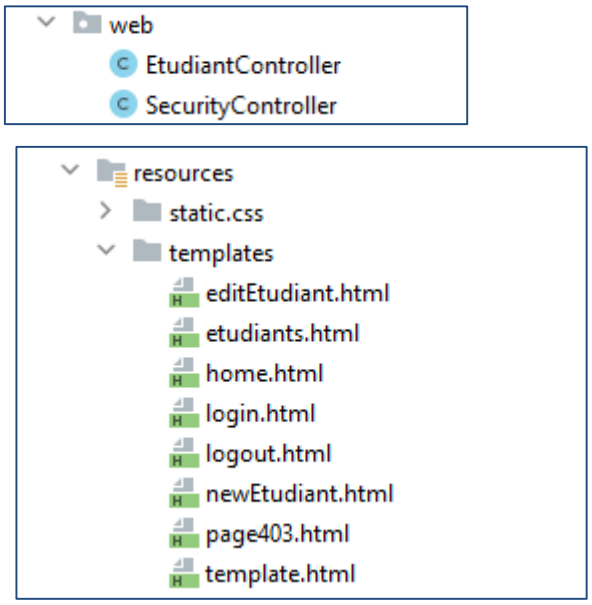
EtudiantRepository.java

```java
package ma.enset.etudiant_ap.repositories;

import ...

@Repository
public interface EtudiantRepository extends JpaRepository<Etudiant, Long> {
    Page<Etudiant> findByNomContains(String keyword, Pageable pageable);
}
```

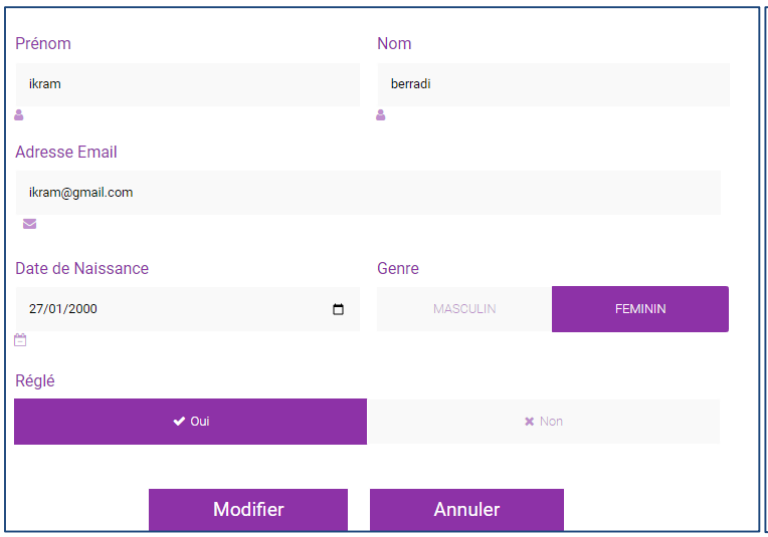
➢ Test de la couche DAO

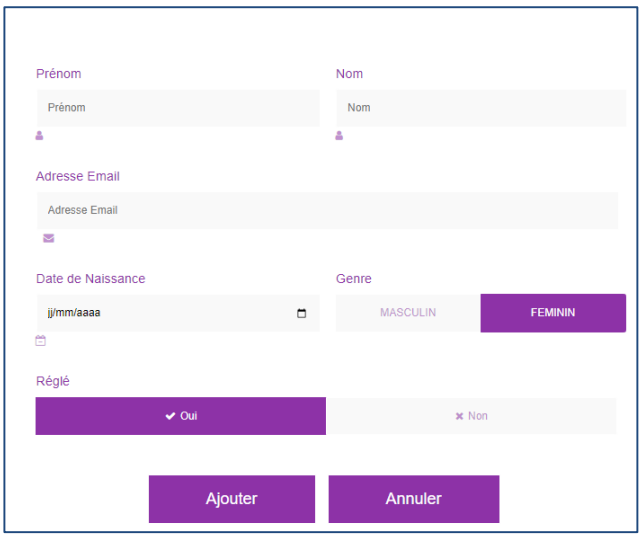
id	date_naissance	email	genre	nom	prenom	regle
1	2000-01-27	ikram@gmail.com	FEMININ	berradi	ikram	1
2	2001-02-07	sara@gmail.com	FEMININ	badaoui	sara	0
3	2000-04-20	salma@gmail.com	MASCULIN	alami	salma	1
4	2000-01-22	mohammed@gmail.com	MASCULIN	alaoui	Mohammed	1

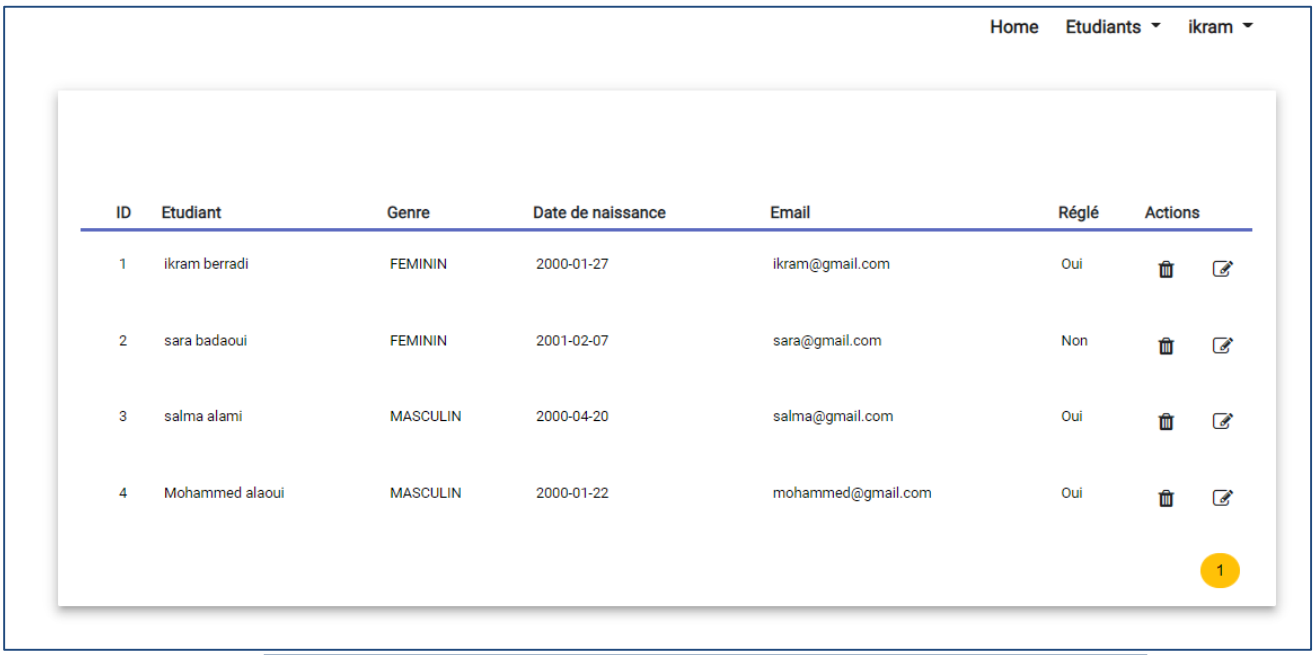
➢ Couche Web

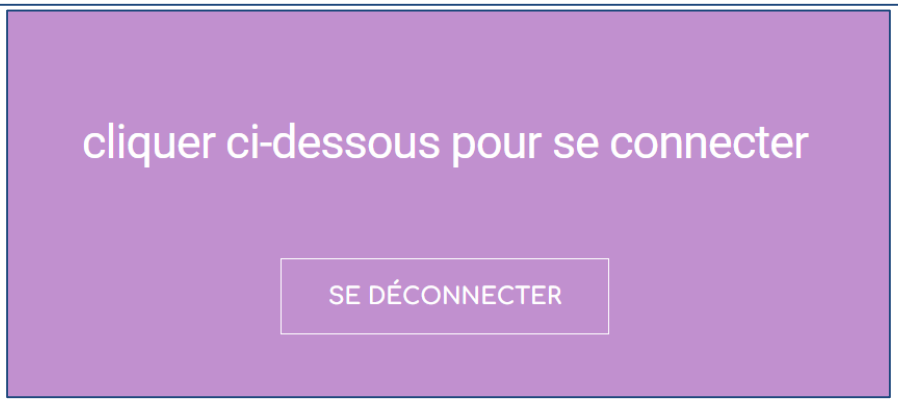


➢ interface


Prénom
ikram
Nom
berradi
Adresse Email
ikram@gmail.com
Date de Naissance
27/01/2000
Genre
MASCULIN
FEMININ
Réglé
Oui
Non
Modifier
Annuler


Prénom
Nom
Adresse Email
Date de Naissance
jj/mm/aaaa
Genre
MASCULIN
FEMININ
Réglé
Oui
Non
Ajouter
Annuler


Home
Etudiants
ikram
ID
Etudiant
Genre
Date de naissance
Email
Réglé
Actions
1 ikram berradi FEMININ 2000-01-27 ikram@gmail.com Oui
2 sara badaoui FEMININ 2001-02-07 sara@gmail.com Non
3 salma alami MASCULIN 2000-04-20 salma@gmail.com Oui
4 Mohammed alaoui MASCULIN 2000-01-22 mohammed@gmail.com Oui
1


cliquer ci-dessous pour se connecter
SE DÉCONNECTER

- **Contrôleur**

EtudiantController.java

```java
@Controller
@AllArgsConstructor
public class EtudiantController
{
    private EtudiantRepository etudiantRepository;
    @GetMapping(path = "/user/index")
    public String etudiants(Model model,
                            @RequestParam(name = "page", defaultValue = "0") int page,
                            @RequestParam(name = "size", defaultValue = "5") int size,
                            @RequestParam(name = "keyword", defaultValue = "") String keyword)
    {
        Page<Etudiant> etudiantPage = etudiantRepository.findByNomContains(keyword, PageRequest.of
        model.addAttribute( attributeName: "listEtudiants", etudiantPage.getContent());
        model.addAttribute( attributeName: "pagesNumber", new int[etudiantPage.getTotalPages()]);
        model.addAttribute( attributeName: "currentPage",page);
        model.addAttribute( attributeName: "keyword", keyword);
        return "etudiants";
    }

    @GetMapping("/admin/delete")
    public String deleteFunction(Long id,
                                 @RequestParam(defaultValue = "") String keyword,
                                 @RequestParam(defaultValue = "0") int page){
        etudiantRepository.deleteById(id);
        return "redirect:/user/index?page="+page+"&keyword="+keyword;
```

```java
    @GetMapping("/")
    public String home() { return "home"; }

    @GetMapping("/admin/newEtudiant")
    public String newEtudiant(Model model)
    {
        model.addAttribute( attributeName: "etudiant", new Etudiant());
        return "newEtudiant";
    }

    @PostMapping(path = "/admin/save")
    public String save(Model model,  @Valid Etudiant etudiant, BindingResult bindingResult,
                       @RequestParam(defaultValue = "") String keyword,
                       @RequestParam(defaultValue = "0") int page)
    {
        //valeur par defaut @RequestParam
        if(bindingResult.hasErrors())
        {
            if(etudiant.getId()==null)
                return "newEtudiant";
            else
                return "editEtudiant";
        }
        etudiantRepository.save(etudiant);
        return "redirect:/user/index?page="+page;
```

```java
            if(etudiant.getId()==null)
                return "newEtudiant";
            else
                return "editEtudiant";
        }
        etudiantRepository.save(etudiant);
        return "redirect:/user/index?page="+page;
    }

    @GetMapping("/admin/editEtudiant")
    public String editEtudiant(Model model, Long id, String keyword, int page){
        Etudiant etudiant=etudiantRepository.findById(id).orElse( other: null);
        if(etudiant==null) throw new RuntimeException("Etudiant introuvable!!!!");
        model.addAttribute( attributeName: "etudiantEdit",etudiant);
        model.addAttribute( attributeName: "page",page);
        model.addAttribute( attributeName: "keyword",keyword);
        return "editEtudiant";
    }

    @GetMapping(value="/login")
    public String login() { return "login"; }

    @GetMapping(value="/logout")
    public String logout() { return "logout"; }
}
```

- **Utilisation des Layouts :**

Templates Généralement toutes les page d'une application web partagent le même contenu html (Header, footer, menus, etc..) Pour éviter des faire des copies coller dans toutes les pages, il est important de définit une page template **(navBar.html)**.
qui définit toutes les parties fixes de toutes les pages (header, footer, menus, etc…)
et déclarer les sections qui changeront de contenue en fonction de chaque page.

- **Vue : etudiant.html**

etudiants.html

```html
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org"
      xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"
      xmlns:sec="http://www.thymeleaf.org/extras/spring-security"
      layout:decorate="template">
<head>
    <meta charset="UTF-8">
    <title>GEtude</title>
    <link rel="stylesheet" th:href="@{/css/styleEtudiants.css}">
    <link rel='stylesheet' href='https://code.getmdl.io/1.3.0/material.indigo-pink.min.css'>
    <link href="https://fonts.googleapis.com/css?family=Roboto:300,400&display=swap" rel="stylesheet">
</head>
<body>

<div layout:fragment="content">
    <div class="mdl-card mdl-card-mediakit mdl-shadow--8dp" style="...">
        <div class="mdl-card__title">
                <h2 class="mdl-card__title-text">Liste des étudiants</h2>
                <!-- searchBar -->
            <form method="get" th:action="@{/user/index}" style="...">
                <div class="search">
                    <input id="searchBar" class="searchbar" type="text" placeholder="Recherhcer . . .
                    <button type="submit" id="btnSearch" class="btn-search"><i class="fa fa-search" s
                </div>
```

- **Vue : editEtudiant.html**

editEtudiant.html

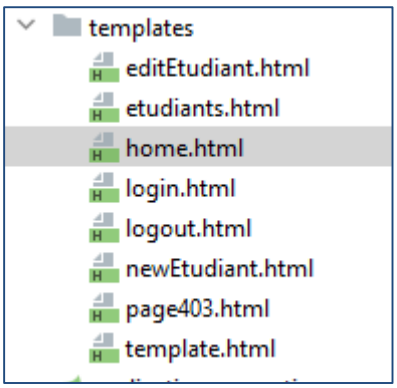
```
        xmlns:sec="http://www.thymeleaf.org/extras/spring-security"
        layout:decorate="template">
<head>
    <meta charset="UTF-8">
    <title>GEtude</title>
    <link rel="stylesheet" th:href="@{/css/styleEtudiants.css}">
    <link rel="stylesheet" th:href="@{/css/styleNewEtudiant.css}">
    <link rel='stylesheet' href='https://code.getmdl.io/1.3.0/material.indigo-pink.min.css'>
    <link href="https://fonts.googleapis.com/css?family=Roboto:300,400&display=swap" rel="stylesheet"
    <style>
        h4{
            font-size: 20px;
            margin: 30px 0 ópx 0;
        }
        .input-group{
            margin:0 0;
        }
    </style>
</head>
<body>

<div layout:fragment="content">
    <div class="mdl-card mdl-card-mediakit mdl-shadow--8dp" style="...">
        <div class="mdl-card__title" style="...">
                <h2 class="mdl-card__title-text" style="...">ID :  <b th:text="${etudiantEdit.id}" st
```

- **Vue : home.html**

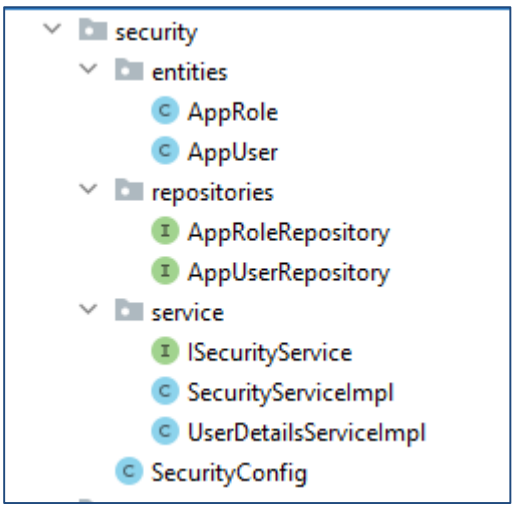
home.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>GEtude</title>
    <link rel="stylesheet" th:href="@{/css/styleHome.css}">
    <link rel='stylesheet' href='https://code.getmdl.io/1.3.0/material.indigo-pink.min.css'>
    <link href="https://fonts.googleapis.com/css?family=Roboto:300,400&display=swap" rel="stylesheet">
</head>
<body>

    <div class="row" id="welcome">
        <h1 class="home-title">
            <span>Bienvenue<b></b> </span>
            <span></span>
        </h1>
    </div>
    <div class="row"></div>
    <div class="row">
        <div class="panel">
            <a class="goHome" th:href="@{/user/index}">Entrer</a>
        </div>
    </div>

```

- **Autres vues**



➢ SPRING SECURITY



- **SecurityConfig.java**

SecurityConfig.java

```java
package ma.enset.etudiant_ap.security;

import ...

@Configuration
@EnableWebSecurity //va etre instancié en premier
public class SecurityConfig extends WebSecurityConfigurerAdapter
{
    @Autowired
    private DataSource dataSource ;
    @Autowired
    private UserDetailsServiceImpl userDetailsService;
    @Autowired
    private PasswordEncoder passwordEncoder;
    //pour spécifier les users
    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws Exception
    {
        //Sécuriser l'accès à l'application avec un système d'authentification basé sur
        // Spring security en utilisant la stratégie UseDetails Service
        auth.userDetailsService(userDetailsService);
    }

    //pour spécifier les droit d'accès de chaque user
    @Override
```

```java
    //pour spécifier les droit d'accès de chaque user
    @Override
    protected void configure(HttpSecurity http) throws Exception
    {
        http.formLogin().loginPage("/login").permitAll().and().logout().logoutUrl("/logout").logoutSuccessUrl("/

        http.authorizeRequests().antMatchers( ...antPatterns: "/").permitAll();
        http.authorizeRequests().antMatchers( ...antPatterns: "/admin/**").hasAuthority("ADMIN");
        http.authorizeRequests().antMatchers( ...antPatterns: "/user/**").hasAuthority("USER");
        http.authorizeRequests().antMatchers( ...antPatterns: "/webjars/**").permitAll();
        http.authorizeRequests().antMatchers( ...antPatterns: "/css/**").permitAll();

        http.authorizeRequests().anyRequest().authenticated();
        // http.authorizeHttpRequests().anyRequest().authenticated();//toutes les requetes http nécissites une au

        http.exceptionHandling().accessDeniedPage("/403"); // page 403
    }

}
```

- Users

active	password	username
1	$2a$10$7razfELso6EsmfM.E.r4n.HjTxn6pmHFHtmNY0BA5.o...	ikram
1	$2a$10$tdHfweAo1FVTpSVqBrdvzeVkTJYtC2dOYI4moZYxEI8...	sara

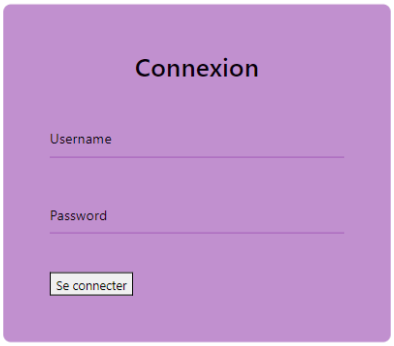
- Roles

role_id	description	role_name
1		ADMIN
2		USER

- Role_Users

app_roles_role_id	users_id
1	10dfe08f-06ee-428e-aeae-e2af96a9cd77
2	10dfe08f-06ee-428e-aeae-e2af96a9cd77
2	710b1557-6e7d-4170-bfed-f9ef5ded69b6

- **Interfaces de Spring Security de l'authentification**
  - **Sign in**



  - **Log out**

➢ Structure finale de projet :