

# Activité Pratique Spring MVC l'hygiène, Spring Sécurité Gestion des Etudiants



Réaliser par :

Mohamed amine

KHAMMOUR

Professeur :

Mr. Mohamed

YOUSSEF

## Enonce d'activité



### Activité Pratique Spring MVC, Spring Data JPA, Spring Security

Mohamed YOUSSEFI • 11 avr.

100 points

Créer une application Web basée sur Spring MVC, Spring Data JPA et Spring Security qui permet de gérer des étudiants.

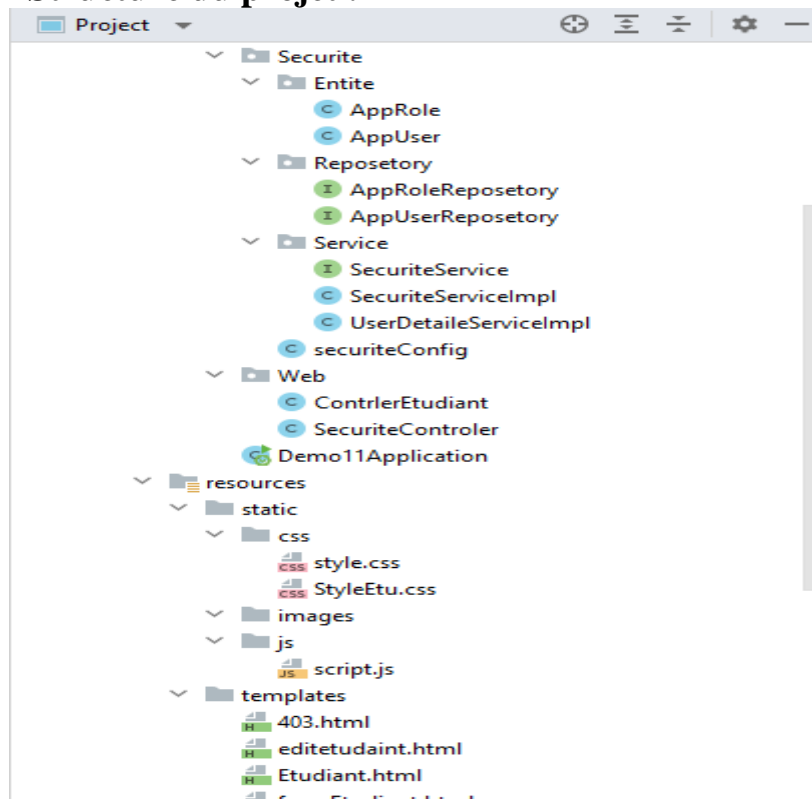
Chaque étudiant est défini par:

- 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 UserDetails 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/TES0?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

```
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 :

```
<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 EtudiantRepository : contient le prototype de la fonction qui permet de retourner un patient par son nom :**

```
import ...
```

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

Le Repository Web qui contient la class ControllerEtudiant :

```
public class ControllerEtudiant {  
    private EtudiantRepository etudiantRepository;  
  
    @GET(path = "/user/index")  
    public String patientes(Model model ,  
        @RequestParam(name = "page", defaultValue = "0") int page,  
        @RequestParam(name = "size", defaultValue = "5") int size,  
        @RequestParam(name = "keyword", defaultValue = "") String keyword)  
    {  
        Page<Etudiant> pagePatient = etudiantRepository.findByNomContains(keyword, PageRequest.of(page, size));  
        model.addAttribute("PageEudiant", pagePatient.getContent());  
        model.addAttribute("nombrePages", new int[] {pagePatient.getTotalPages()});  
        model.addAttribute("currentPage", page);  
        model.addAttribute("keyword", keyword);  
        return "Etudiant.html";  
    }  
  
    @GET(path = "/Admin/delete")  
    public String delete(long id, String keyword, int page){  
        etudiantRepository.deleteById(id);  
    }  
}
```

- Dans le chemin `ressources/templates`, j'ai créé le template `Etudiant.html`, et d'autres templates sont traités par le moteur de rendu `thymeleaf`, reçoivent les données nécessaires à travers le contrôleur.
- Dans ce template j'ai inclus les frameworks `bootstrap` et `jquery` qui sont stockés 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:${PageEtudiant}">
<td th:text="${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="${p.genre}"></td>
<td sec:authorize="hasAuthority('ADMIN')">
<a onclick="return confirm('Etes vous sure?')" class="btn btn-danger" th:href="@{/Admin/delete(id
Delete
</a>
</td>
<td sec:authorize="hasAuthority('ADMIN')">
<a class="btn btn-success" th:href="@{/Admin/editEtudiant(id=${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 verifyPassword);
    AppRole saveNewRole(String roleName,String description);
    void addRoleToUser(String username,String roleName);
    void removeRoleToUser(String username,String roleName);
    AppUser loadUserByUsername(String username);
}
```

Extrais de l'implémentation de l'interface :

```
@Transactional
public class SecuriteServiceImpl implements SecuriteService {
    AppRoleRepository appRoleRepository;
    AppUserRepository appUserRepository;
    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=appUserRepository.save(appUser);
        return savedAppUser;
    }
}
```

## La class UserDetailsServiceImpl

```
@Service
@AllArgsConstructor
public class UserDetailsServiceImpl 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().stream()
                .map(role->new SimpleGrantedAuthority(role.getRole()))
                .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;
    @UserDetailsService @Autowired
    private UserDetailsService userDetailsService;
    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws Exception {
        auth.userDetailsService(userDetailsService);
        /*
```

```
        @Override
        protected void configure(HttpSecurity http) throws Exception {
            http.formLogin().loginPage("/login").permitAll(); // default login page
            http.csrf().disable();

            http.authorizeRequests().antMatchers("/Admin/**").hasAuthority("ADMIN");
            http.authorizeRequests().antMatchers("/user/**").hasAuthority("USER");
            http.authorizeRequests().antMatchers("/webjars/**", "/js/**", "/css/**", "/images/**", "/resources/**").permitAll();
            http.authorizeRequests().antMatchers("/").permitAll();

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

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

## La class application :

```
@SpringBootApplication
public class Demo1Application {

    public static void main(String[] args) { SpringApplication.run(Demo1Application.class, args); }

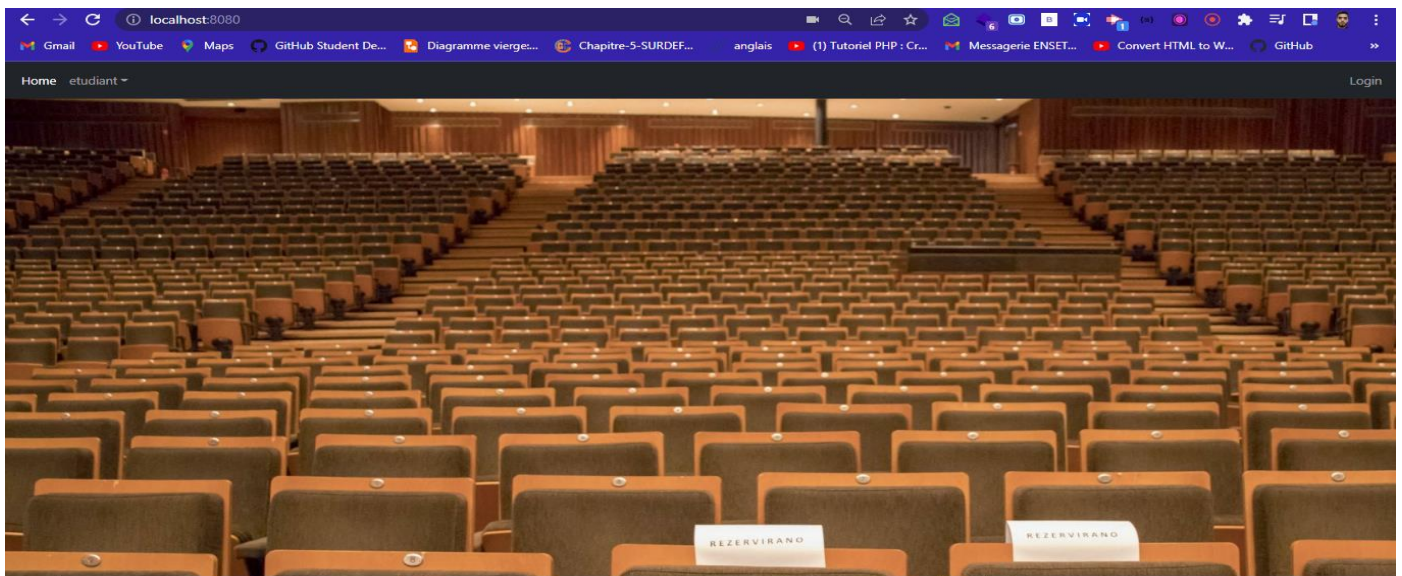
    // @Bean
    CommandLineRunner commandLineRunner(EtudiantRepository etudiantRepository){
        return args -> {
            etudiantRepository.save(new Etudiant(id: null, nom: "amine", Prenom: "khammour", new Date(), EnRegle: false, StatueGenre.F));
            etudiantRepository.save(new Etudiant(id: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

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

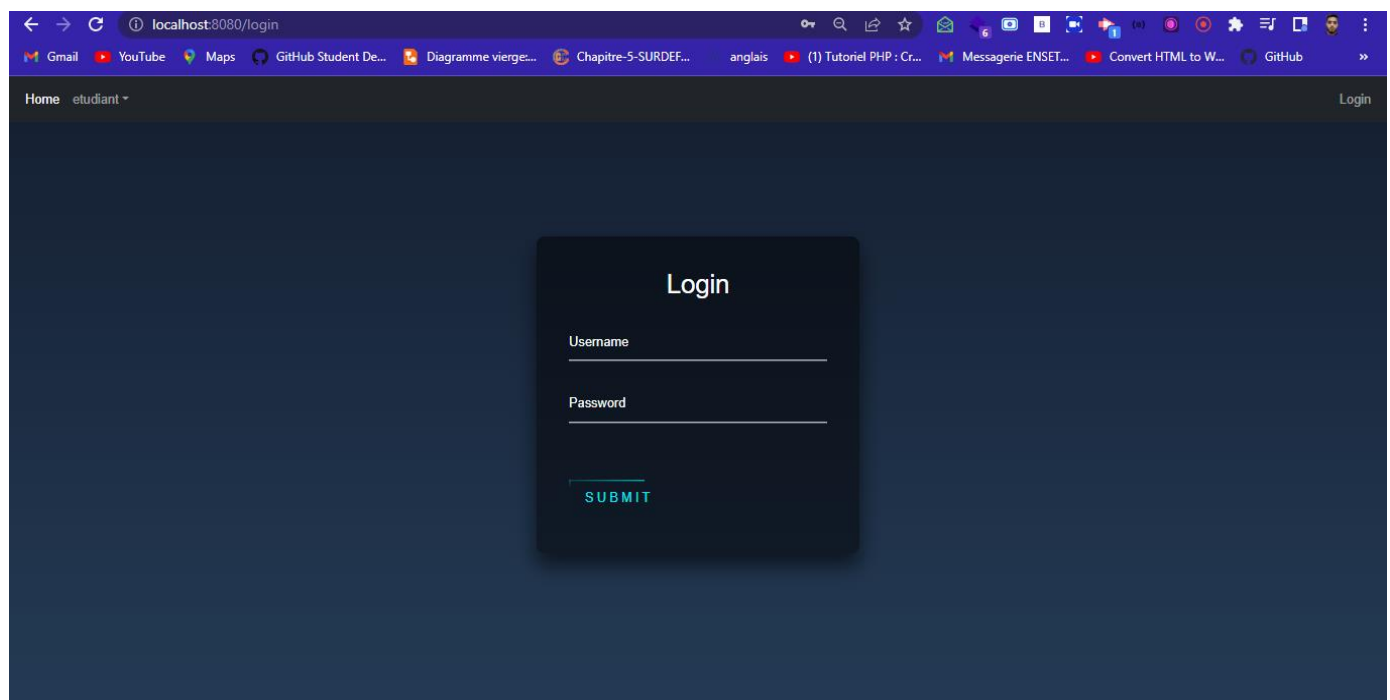


```
//@Bean
CommandLineRunner SaveUser(SecuriteService securiteService){
    return args -> {
        securiteService.saveNewUser( username: "mohamed", password: "1234", verifyPasswor: "1234");
        securiteService.saveNewUser( username: "yassmin", password: "1234", verifyPasswor: "1234");
        securiteService.saveNewUser( username: "hassan", password: "1234", verifyPasswor: "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 :



## La page de login :





## User Admin :

localhost:8080/user/index?page=19&keyword=

Home etudiant Search search mohamed

### LISTES DES ETUDIANT

ID	NOM	PRENOM	DATE	ENREGLE	GENRE	EDITE	DELET
160	hatim	prenom	2022-04-17	false	M	Delete	Edite
161	rafik	prenom	2022-04-17	true	M	Delete	Edite
162	khamour	khamour	2022-04-01		M	Delete	Edite
164	khalid	prenom	2022-04-17	true	F	Delete	Edite

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21

## User Normal :

localhost:8080/user/index?page=18&keyword=

Home etudiant Search search yassmin

### LISTES DES ETUDIANT

ID	NOM	PRENOM	DATE	ENREGLE	GENRE
153	rafik	prenom	2022-04-17	true	M
155	khalid	prenom	2022-04-17	true	F
156	hatim	prenom	2022-04-17	false	M
157	rafik	prenom	2022-04-17	true	M
159	khalid	prenom	2022-04-17	true	F

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21

## La partie d'ajout :

The screenshot shows a web browser at the URL `localhost:8080/Admin/formeEtudiant`. The page has a dark blue header with a navigation bar containing 'Home' and 'etudiant' with a dropdown arrow. A search bar is on the right of the header. The main content area has a dark blue background with the title 'Ajouter un Etudiant' in a large, light blue font. Below the title, there is a form with the following fields: 'nom' (text input), 'prenom' (text input), 'date d'inscription' (date picker showing 'mm/dd/yyyy'), a checkbox labeled 'en régie', a label 'Sexe de l\'etudiant :', and a dropdown menu 'Choisir le genre...'. At the bottom of the form is a blue button labeled 'Enregistrer'.

## La partie modification :

The screenshot shows a web browser at the URL `localhost:8080/Admin/editEtudiant?id=12&keyword=&page=0`. The page has a dark blue header with a navigation bar containing 'Home' and 'etudiant' with a dropdown arrow. A search bar is on the right of the header. The main content area has a dark blue background with the title 'Modifier un Etudiant' in a large, light blue font. Below the title, there is a form with the following fields: 'nom' (text input with value 'rafik'), 'prenom' (text input with value 'prenom'), 'date d'inscription' (date picker showing '04/17/2022'), a checkbox labeled 'en régie', a label 'Sexe de l\'etudiant :', and a dropdown menu 'Choisir le genre...'. At the bottom of the form is a blue button labeled 'Enregistrer'.

La page d'erreurs :

