

Aida Magou Diop

M1 GLSI

Cours Inf3522 - Développement d'Applications JEE

Lab 5 : Sécurisation

Ajoutons les dépendances nécessaires:

```
dependencies {  
    implementation 'org.springframework.boot:spring-boot-starter-web'  
    implementation 'org.springframework.boot:spring-boot-starter-data-jpa'  
    implementation 'org.springframework.boot:spring-boot-starter-data-rest'  
    implementation group: 'org.springdoc', name: 'springdoc-openapi-starter'  
    developmentOnly 'org.springframework.boot:spring-boot-devtools'  
    testImplementation 'org.springframework.boot:spring-boot-starter-test'  
    runtimeOnly 'org.mariadb.jdbc:mariadb-java-client'  
  
    implementation 'org.springframework.boot:spring-boot-starter-security'  
    testImplementation 'org.springframework.security:spring-security-test'  
}
```

AppUser.java :

```
> java > fst > dmi > cardatabase > domain > J AppUser.java > Language Support for Java  
1  package fst.dmi.cardatabase.domain;  
2  
3  import jakarta.persistence.Column;  
4  import jakarta.persistence.Entity;  
5  import jakarta.persistence.GeneratedValue;  
6  import jakarta.persistence.GenerationType;  
7  import jakarta.persistence.Id;  
8  // import lombok.Data;  
9  
10 @Entity  
11 public class AppUser {  
12     @Id  
13     @GeneratedValue(strategy = GenerationType.AUTO)  
14     private Long id;  
15  
16     @Column(nullable = false, unique = true)  
17     private String username;  
18  
19     @Column(nullable = false)  
20     private String password;  
21  
22     @Column(nullable = false)  
23     private String role;  
24
```

Activer W
Accédez aux

UserDetailsServiceImpl.java :

```
AppUser.java U × J AppUserRepository.java U J UserDetailsServiceImpl.java U × ▶ ~ ⓘ
src > main > java > fst > dmi > cardatabase > service > J UserDetailsServiceImpl.java
1 package fst.dmi.cardatabase.service;
2
3 // import org.springframework.security.core.userdetails.*;
4 import org.springframework.security.core.userdetails.User;
5 import org.springframework.security.core.userdetails.UserDetails;
6 import org.springframework.security.core.userdetails.UserDetailsService;
7 import org.springframework.security.core.userdetails.UsernameNotFoundException;
8 import org.springframework.stereotype.Service;
9 import fst.dmi.cardatabase.domain.AppUser;
10 import fst.dmi.cardatabase.domain.AppUserRepository;
11
12 @Service
13 public class UserDetailsServiceImpl implements UserDetailsService {
14
15     private final AppUserRepository repository;
16
17     public UserDetailsServiceImpl(AppUserRepository repository) {
18         this.repository = repository;
19     }
20
21     @Override
```

Modifions la sécurité pour utiliser la base :

SecurityConfig.java :

```
J SecurityConfig.java 8. U × application.properties M J AppUser.java U J AppUserRepository.java U J ▶ ~ ⓘ
src > main > java > fst > dmi > cardatabase > J SecurityConfig.java > Language Support for Java(TM) by Red Hat > SecurityConfig > config
10 import org.springframework.security.provisioning.InMemoryUserDetailsManager;
11 import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
12 import org.springframework.security.crypto.password.PasswordEncoder;
13
14 import fst.dmi.cardatabase.service.UserDetailsServiceImpl;
15
16
17 @Configuration
18 @EnableWebSecurity
19 public class SecurityConfig {
20     private final UserDetailsServiceImpl userDetailsService;
21
22     public SecurityConfig(UserDetailsServiceImpl userDetailsService) {
23         this.userDetailsService = userDetailsService;
24     }
25
26     @Bean
27     public AuthenticationManager authenticationManager(AuthenticationConfiguration authConfig) throws Exception {
28         return authConfig.getAuthenticationManager();
29     }
30
31     @Bean
32     public PasswordEncoder passwordEncoder() {
33         return new BCryptPasswordEncoder();
34     }
35 }
```

Enregistrons des utilisateurs test dans CardatabaseApplication.java

```
Untitled-1  J CardatabaseApplication.java 4, M  J AppUserRepository.java U
CardatabaseApplication.java > Language Support for Java(TM) by Red Hat > CardatabaseApplication > runner(CarRepository, OwnerRepository, A
20 public class CardatabaseApplication {
32     return args -> {
33         // Ajouter des propriétaires
34         Owner owner1 = new Owner(firstname:"John", lastname:"Johnson");
35         Owner owner2 = new Owner(firstname:"Mary", lastname:"Robinson");
36         orepository.saveAll(Arrays.asList(owner1, owner2));
37
38         // Ajouter des voitures
39         repository.save(new Car(brand:"Ford", model:"Mustang", color:"Rouge", registrationNu
40         repository.save(new Car(brand:"Nissan", model:"Leaf", color:"Blanc", registrationNu
41         repository.save(new Car(brand:"Toyota", model:"Prius", color:"Argent", registration
42
43         for (Car car : repository.findAll()) {
44             logger.info(car.getBrand() + " " + car.getModel());
45         }
46
47         // Ajouter des utilisateurs avec des mots de passe hachés
48         urepository.save(new AppUser(username:"user",
49             password:"$2a$10$NVM0n8ElaRgg7zW01CxUdei7vWoPg91Lz2aYavh9.f9q0e4bRadue",
50             role:"USER"));
51
52         urepository.save(new AppUser(username:"admin",
53             password:"$2a$10$8cjz47bjbR4Mn8GMg9IZx.vyjhLXR/SKKMSZ9.mP9vpMu0ssKi8GW",
54             role:"ADMIN"));
55     };
56 }
57 }
```

Les utilisateurs apparaissent bien dans la BD :

```
MariaDB [cardb]> show tables;
+-----+
| Tables_in_cardb |
+-----+
| app_user         |
| app_user_seq     |
| car              |
| car_seq          |
| owner            |
| owner_seq        |
+-----+
6 rows in set (0.002 sec)

MariaDB [cardb]> select * from app_user;
+-----+-----+-----+-----+
| id | password | role | username |
+-----+-----+-----+-----+
| 1 | $2a$10$NVM0n8ElaRgg7zW01CxUdei7vWoPg91Lz2aYavh9.f9q0e4bRadue | USER | user |
| 2 | $2a$10$8cjz47bjbR4Mn8GMg9IZx.vyjhLXR/SKKMSZ9.mP9vpMu0ssKi8GW | ADMIN | admin |
+-----+-----+-----+-----+
2 rows in set (0.000 sec)
```

Testons avec Postman :

HTTP **http://localhost:8081/login** Save

POST ▼ **http://localhost:8081/login** Send ▼

Params Authorization Headers (9) **Body** Pre-request Script Tests Settings Cookies

● none ● form-data ● x-www-form-urlencoded ● raw ● binary **JSON** ▼ Beautify

```
1 {
2   "username": "user",
3   "password": "user"
4 }
```

Body Cookies (1) **Headers (11)** Test Results 401 Unauthorized 5 ms 342 B Save Response ▼

Key	Value
X-Content-Type-Options ⓘ	nosniff
X-XSS-Protection ⓘ	0
Cache-Control ⓘ	no-cache, no-store, max-age=0, must-revalidate
Pragma ⓘ	no-cache
Expires ⓘ	0
X-Frame-Options ⓘ	DENY

Testons sur le navigateur :

localhost:8081/login

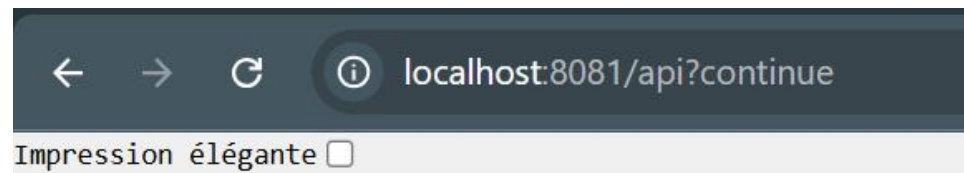
Please sign in

user

....

Sign in

Avec l'authentification, nous pouvons voir que la réponse contient nos ressources API, comme illustré dans la capture d'écran suivante :



```
{
  "_links" : {
    "owners" : {
      "href" : "http://localhost:8081/api/owners"
    },
    "cars" : {
      "href" : "http://localhost:8081/api/cars"
    },
    "appUsers" : {
      "href" : "http://localhost:8081/api/appUsers"
    },
    "profile" : {
      "href" : "http://localhost:8081/api/profile"
    }
  }
}
```