Amina Dahmouni

Classe : II-BDCC2

[Compte Rendu]

Use Case JPA Hibernate Spring Data
Many To Many Case
Users et Roles

Architecture JEE et Middlewares

L'Ecole Normale Supérieure de l'Enseignement Technique de Mohammedia (ENSET)

Les entités :

- **Classe User**

User.java

```java
import javax.persistence.*;
import java.util.ArrayList;
import java.util.List;
@Entity
@Data @NoArgsConstructor @AllArgsConstructor
public class User {
    @Id
    private String userId;
    @Column(name = "USER_NAME",unique = true,length = 20)
    private String username;
    @JsonProperty(access = JsonProperty.Access.WRITE_ONLY)
    private String password;
    //Charger les roles de chaque user
    @ManyToMany(mappedBy = "users",fetch = FetchType.EAGER)
    /*Dons il faut inicialiser la liste pour
      éviter un nullPointerException
     */
    private List<Role> roles=new ArrayList<>();
}
```

- **Classe Role**

Role.java

```java
import javax.persistence.*;
import java.util.ArrayList;
import java.util.List;
@Entity
@Data @NoArgsConstructor @AllArgsConstructor
public class Role {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @Column(name = "DESCRIPTION")
    private  String desc;
    @Column(unique = true,length = 20)
    private String roleName;
    @ManyToMany(fetch = FetchType.EAGER)
    //@JoinTable(name = "USERS_ROLES")
    @ToString.Exclude
    //ne  pas inclure la liste des users d'un role  dans la méthode toString
    @JsonProperty(access = JsonProperty.Access.WRITE_ONLY)
    private List<User> users=new ArrayList<>();
}
```

Repositories :

- **UserRepository**

UserRepository.java

```java
package ma.enst.jpaenset.repositories;
import ma.enst.jpaenset.entities.User;
import org.springframework.data.jpa.repository.JpaRepository;

public interface UserRepository extends JpaRepository<User,String> {
    User findByUsername(String userName);
}
```

- **RoleRepository**

RoleRepository.java

```java
package ma.enst.jpaenset.repositories;
import ma.enst.jpaenset.entities.Role;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

//@Repository pour indiquer que c'est un componenet de la couche dao
public interface RoleRepository extends JpaRepository<Role,Long> {
    Role findByRoleName(String roleName);
}
```

Couche métier :

- **UserService**

UserService.java

```java
package ma.enst.jpaenset.services;
import ma.enst.jpaenset.entities.Role;
import ma.enst.jpaenset.entities.User;
public interface UserService {
    User addNewUser(User user);
    Role addNewRole(Role role);
    User findUserByUserName(String userName);
    Role findRoleByRoleName(String roleName);
    void addRoleToUser(String username,String rolename);
    User authentificate(String userName,String password);
}
```

- **L'implémentation de UserService**

UserServiceImpl.java

```java
package ma.enst.jpaenset.services;
import ...
@Service
@Transactional
@AllArgsConstructor
public class UserServiceImpl implements UserService {
    private UserRepository userRepository;
    private RoleRepository roleRepository;
    /*public UserServiceImpl(UserRepository userRepository, RoleRepository roleRepository) {
        this.userRepository = userRepository;
        this.roleRepository = roleRepository;
    }*/
    /*Si on veut faire l'injection des dépendances via la constructeur
      il faut faire un seule constructeur avec paramètre
     */
    @Override
    public User addNewUser(User user) {
        /**
         *Avec UUID à chaque fois qu'on génére un Id il est unique et
          qui depende de la date système
         */
        user.setUserId(UUID.randomUUID().toString());
        return userRepository.save(user);
    }
```

```java
    @Override
    public Role addNewRole(Role role) {
        return roleRepository.save(role);
    }
    @Override
    public User findUserByUserName(String userName) {
        return userRepository.findByUsername(userName);
    }
    @Override
    public Role findRoleByRoleName(String roleName) {
        return roleRepository.findByRoleName(roleName);
    }
```

```java
    @Override
    public void addRoleToUser(String username, String rolename) {
        User user=findUserByUserName(username);
        Role role=findRoleByRoleName(rolename);
        if (user.getRoles()!=null){
            user.getRoles().add(role);
            /**C'est nessecaire de faire userRepository.save(user)
             * car déja la méthode est transactionelle
             * donc tt les changement sont attribués dans la BD
             */
            //userRepository.save(user)
            role.getUsers().add(user);
        }
    }
    @Override
    public User authentificate(String userName, String password) {
        User user=userRepository.findByUsername(userName);
        if (user==null) throw new RuntimeException("Bad credentials");
        if (user.getPassword().equals(password)) {
            return user;
        }
        throw  new RuntimeException("Bad credentials");
    }
}
```

Application.java :

Création des utilisateurs➔

JpaEnsetApplication.java

```java
package ma.enst.jpaenset;
import ...
@SpringBootApplication
public class JpaEnsetApplication {
    public static void main(String[] args) {
        SpringApplication.run(JpaEnsetApplication.class, args);
    }
    @Bean
    CommandLineRunner start(UserService userService){
          return  args -> {
              User u1=new User();
              u1.setUsername("user1");
              u1.setPassword("123456");
              userService.addNewUser(u1);

              User u2=new User();
              u2.setUsername("admin");
              u2.setPassword("123456");
              userService.addNewUser(u2);
```

Création des rôles et l'ajout des rôles à des utilisateurs➔

```
29          Stream.of("STUDENT","USER","ADMIN").forEach(r->{
30              Role role1=new Role();
31              role1.setRoleName(r);
32              userService.addNewRole(role1);
33          });
34          userService.addRoleToUser( username: "user1", rolename: "STUDENT");
35          userService.addRoleToUser( username: "user1", rolename: "USER");
36          userService.addRoleToUser( username: "admin", rolename: "USER");
37          userService.addRoleToUser( username: "admin", rolename: "ADMIN");
38
```

Dans la base de données➔

Table user :

user_id	password	user_name
27b982b0-1d58-4eb4-b488-d9f0b6ba2f17	123456	admin
e9d2186a-e65e-4460-a286-a1a7a61396fb	123456	user1

Table role :

id	description	role_name
1	*NULL*	STUDENT
2	*NULL*	USER
3	*NULL*	ADMIN

Table d'association :

roles_id	users_user_id
1	e9d2186a-e65e-4460-a286-a1a7a61396fb
2	e9d2186a-e65e-4460-a286-a1a7a61396fb
2	27b982b0-1d58-4eb4-b488-d9f0b6ba2f17
3	27b982b0-1d58-4eb4-b488-d9f0b6ba2f17

L'authentification➔

```
        try {
            User user =userService.authentificate( userName: "user1", password: "123456");
            System.out.println(user.getUserId());
            System.out.println(user.getUsername());
            System.out.println("Roles==>");
            user.getRoles().forEach(r -> {
                System.out.println("Roles==>"+r);
            });
        }
        catch (Exception exception){
            exception.printStackTrace();
        }
```

Les informations de l'utilisateur authentifié :

```
User ID : ee977ee4-2010-486b-9629-24e3623ddfbe
User name : user1
les Roles de ce utilisateur :
*******************************
role==>Role(id=1, desc=null, roleName=STUDENT)
role==>Role(id=2, desc=null, roleName=USER)
```

Si le mot de passe est incorrect :

```
java.lang.RuntimeException Create breakpoint : Bad credentials
```

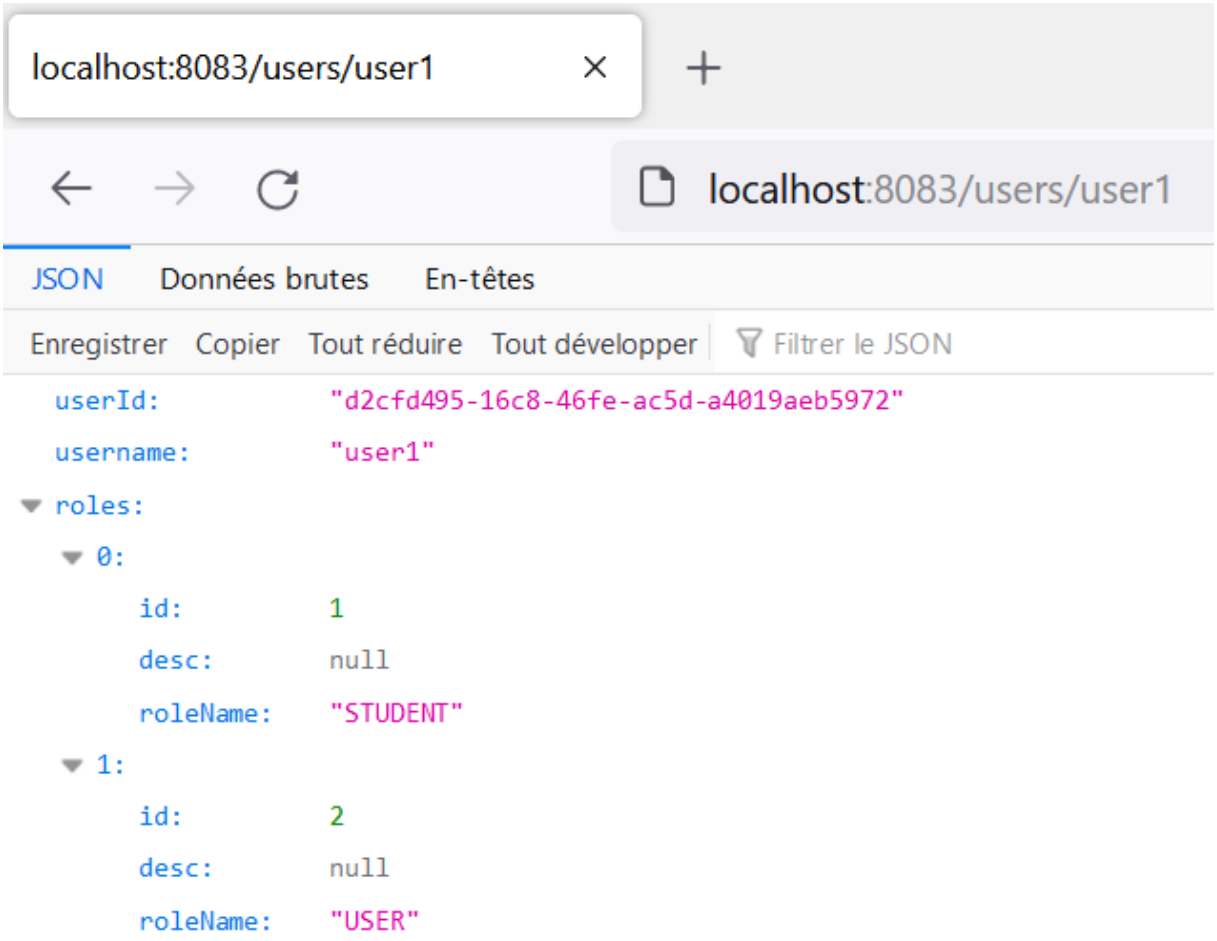
Exception avec le message spécifié dans la fonction authetificate.

Couche web :

UserController.java

```java
package ma.enst.jpaenset.web;
import ma.enst.jpaenset.entities.User;
import ma.enst.jpaenset.services.UserService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RestController;
@RestController
public class UserController {
    @Autowired
    private UserService userService;
    @GetMapping("/users/{username}")
    public User user(@PathVariable String username){
      User user=userService.findUserByUserName(username);
      return user;
    }
}
```

Pour chercher les informations d'un utilisateur avec le mot clé « username ».


localhost:8083/users/user1
localhost:8083/users/user1
JSON
Données brutes
En-têtes
Enregistrer
Copier
Tout réduire
Tout développer
Filtrer le JSON
userId: "d2cfd495-16c8-46fe-ac5d-a4019aeb5972"
username: "user1"
roles:
0:
id: 1
desc: null
roleName: "STUDENT"
1:
id: 2
desc: null
roleName: "USER"